НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації та управління

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
"___"_____2019 р.

Дипломний проект
на здобуття ступеня бакалавра

з напряму підготовки *6.050101 «Комп'ютерні науки»*

на тему: «Система обробки зображень облич людей за допомогою алгоритмів комп'ютерного зору »

Виконала: студентка *4* курсу, групи *ІС-52*
Нго Май Фионг _____
(прізвище, ім'я, по батькові) (підпис)

Керівник *доц., к.т.н., доц. Ковалюк Т.В.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації *ст. вик. Халус О. А.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____
(підпис)

Київ – 2019 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації та управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) *6.050101*

«Комп’ютерні науки» («Інформаційні управляючі системи та технології»)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)
“___”_____2019 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Нго Май Фионг
(прізвище, ім’я, по батькові)

1. Тема проекту «Система обробки зображень облич людей за допомогою алгоритмів комп'ютерного зору»

керівник проекту *Ковалюк Т.В., к.т.н., доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “*23*”*квітня* 2019 р. №1181-с

2. Термін подання студентом проекту “*03*”*червня* 2019 року

3. Вихідні дані до проекту

Технічне завдання

4. Зміст пояснювальної записки

1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі

2. Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних

3. Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв’язання

4. Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів

5. Технологічний розділ: керівництво користувача, методика випробувань

програмного продукту

5. Перелік графічного матеріалу

1. Схема структурна діяльності системи

2. Схема структурна контекстної моделі системи

3. Схема структурна функціональної моделі IDEF0

4. Схема структурна класів програмного забезпечення

5. Схема структурна послідовності системи

6. Схема структурна компонентів системи

7. Креслення виду екранних форм

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-5	*Халус О. А.*		

7. Дата видачі завдання *«23» квітня 2019 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	*Вивчення рекомендованої літератури*	*18.04.2019*	
2.	*Аналіз існуючих методів розв'язання задачі*	*20.04.2019*	
3.	*Постановка та формалізація задачі*	*22.04.2019*	
4.	*Розробка інформаційного забезпечення*	*23.04.2019*	
5.	*Алгоритмізація задачі*	*25.04.2019*	
6.	*Обґрунтування використовуваних технічних засобів*	*05.05.2019*	
7.	*Розробка програмного забезпечення*	*15.05.2019*	
8.	*Налагодження програми*	*19.05.2019*	
9.	*Виконання графічних документів*	*26.05.2019*	
10.	*Оформлення пояснювальної записки*	*29.05.2019*	
11.	*Подання ДП на попередній захист*	*30.05.2019*	
12.	*Подання ДП на основний захист*	*03.06.2019*	
13.	*Подання ДП рецензенту*	*05.06.2019*	

Студент _____ Нго М.Ф.
(підпис)

Керівник проекту _____ Ковалюк Т.В.
(підпис)

№з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
			Документи текстові		
1	А4	ДП ІС-5218.1181-с.ПЗ	Пояснювальна записка	66	
2	А4	ДП ІС-5218.1181-с.ТЗ	Технічне завдання	10	
3	А4	ДП ІС-5218.1181-с.ПМВ	Програма та методика випробувань	10	
			Матеріали графічні		
4	А3	ДП ІС-5218.1181-с.ССД	Схема структурна діяльності системи	1	
5	А3	ДП ІС-5218.1181-с.СКМ	Схема структурна контекстної моделі системи	1	
6	А3	ДП ІС-5218.1181-с.СФМ	Схема структурна функціональної моделі IDEF0	1	
7	А3	ДП ІС-5218.1181-с.СКЗ	Схема структурна класів програмного забезпечення	1	
8	А3	ДП ІС-5218.1181-с.СПС	Схема структурна послідовності системи	1	
9	А3	ДП ІС-5218.1181-с.СКС	Схема структурна компонентів системи	1	
10	А3	ДП ІС-5218.1181-с.КЕФ	Креслення виду екранних форм	1	
					ДП ІС-5218.1181-с.ВДП

З	Арк.	ПІП	Підп.	Дата	ДП ІС-5218.1181-с.ВДП Літ.		
Р	ПІБ	Нго Май Фионг		Від			
Кері	ПІБ керів.	Ковалюк Т.В.		Дата			
Консульт					Відомість дипломного проекту	КПІ ім.	Лист 1 / Листів 1
Н/контр.		Халус О.А.					
В.о.зав.каф.		Павлов О.А.				КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52	

Пояснювальна записка
до дипломного проекту

на тему: «Система обробки зображень облич людей за допомогою алгоритмів комп'ютерного зору»

Київ – 2019 року

АНОТАЦІЯ

Пояснювальна записка дипломного проекту складається з п’яти розділів, містить 97 сторінок, 18 рисунків, 7 таблиць, один додаток та 22 джерел.

Дипломний проект присвячено розробці системи обробки зображень облич людей на прикладі зміни зображень їх емоцій з використанням інструментів комп’ютерного зору та комп’ютерної графіки.

Було використано нейронні мережі, а саме мережі, навчені за методом логістичної регресії, а також алгоритми спотворення візуальних даних.

Розділ інформаційне забезпечення містить інформацію про вхідні та вихідні дані, а також формат даних, які використовуються для тренування нейронної мережі.

Розділ з математичного забезпечення містить математичні формули, які пояснюють принцип роботи логістичної регресії для тренування нейронної мережі, та алгоритму спотворення зображення.

У розділі програмного та технічного забезпечення описано мінімальні вимоги до забезпечення для успішного використання програмного продукту.

У технологічному розділі описується інструкція користувача та тестування роботи системи.

КОМП’ЮТЕРНИЙ ЗІР, НЕЙРОННІ МЕРЕЖІ, ОБРОБКА ЗОБРАЖЕНЬ, ЛОГІСТИЧНА РЕГРЕСІЯ, СПОТВОРЕННЯ ВІЗУАЛЬНИХ ДАНИХ.

ABSTRACT

The thesis consists of five sections, 97 pages, 18 figures, 7 tables, one attachment and 22 references.

The work is dedicated to the research on ways to process facial images taking as an example changing emotions on human's face using computer vision and computer graphics tools.

Among the used algorithms, there are neural networks taught using logistic regression method, as well as visual data liquify algorithm.

The information section provides details on input-output data and data, used to train the neural network.

Math section provides the solutions in the form of math formulas, explaining the principle of logistic regression to train the neural networks and liquify algorithm.

Software and technical requirements section explains the minimum requirements related to software and hardware in order for the system to work properly.

Technological section provides information on how to use the product and the testing process.

COMPUTER VISION, NEURAL NNETWORKS, IMAGE PROCESSING, LOGISTIC REGRESSION, VISUAL DATA DISTORTION.

ЗМІСТ

ВСТУП

Стрімкий розвиток штучного інтелекту спричинив народженню таких напрямків як комп'ютерний зір, а з існуванням комп'ютерів завжди мало сенс розвивати і комп'ютерну графіку. Але не завжди комп'ютерний зір та комп'ютерну графіку намагалися поєднати разом. На сьогодні існує чимало продуктів по обробці зображень об'єктів, які були розпізнані за допомогою інструментів комп'ютерного зору. Але, зазвичай, такі продукти були створені у приватних компаніях, використовуючи групи вчених та розробників, які залишали програмний код конфіденційним. Але чи можемо ми відтворити це за допомогою існуючих інструментів, які перебувають у вільному доступі?

У даній роботі досліджується метод відтворення системи обробки зображень розпізнаних об'єктів на прикладі облич людей. Варто зазначити, що важливо було не використовувати зовнішні візуальні дані для обробки вхідних даних, а також використовувати тільки інструменти розробки, які перебувають у вільному доступі.

Призначенням роботи є створення системи, яка приймає на вхід візуальні дані різних форматів, розпізнає на них обличчя та змінює на ньому емоцію, при цьому не використовуючи елементів зображень з зовнішніх джерел.

Мета вирішення даної проблеми полягає у тому, щоб знайти доступний спосіб генерування контенту з вже існуючого, використовуючи алгоритми штучного інтелекту та комп'ютерного зору, при цьому мінімально змінюючи ключові риси зображення.

Ціллю роботи є спростити обчислювальний процес генерування даних, використовуючи інструменти штучного інтелекту при цьому маючи лише візуальні дані одного типу.

Для досягнення цілі потрібно розв'язати наступні задачі:

- Реалізувати процес генерування нових візуальних даних, який не потребує допоміжні інструменти;

- створити нові методи комбінування інструментів комп’ютерного зору та обробки зображень;
- полегшити маніпулювання візуальними даними;
- полегшити процес розробки систем, які одночасно використовують інструменти комп’ютерного зору та обробки зображень.

Загальний процес розробки складається з двох етапів: розпізнавання самих об’єктів та його спотворення.

Проблему спотворення розпізнаних образів пропонується вирішити за допомогою нейронних мереж для знаходження орієнтирів та деяких інструментів обробки зображень. Спочатку нейронні мережі тренуються для розпізнавання орієнтирів на великій кількості зображень з обличчями у різному ракурсі та різних розмірів.

Після успішного розпізнавання орієнтирів на обличчі пропонується використати алгоритми обробки зображень, такі як хвильові та шарові алгоритми. Вони зміщують деякі частини на об’єкті і таким чином утворюють ефект «зміни емоції».

Робота була опублікована на міжнародній конференції Intsol 2019.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

Навчити машину «бачити» є досить нелегкою задачею. Для того, щоб машина розпізнавала такі об’єкти як люди або тварини, вона звертається до комп’ютерного зору та розпізнавання образів.

Комп’ютерний зір – це наука про розпізнавання, аналіз та обробку об’єктів на зображеннях комп’ютером [1].

Комп’ютерний зір також можна вважати точкою опори для штучного інтелекту. Вона дає змогу комп’ютеру розумно інтерпретувати отримані візуальні дані та класифікувати їх.

Не дивлячись на те, що комп’ютерний зір став популярним лише у 2012, ця наука не є новою. Вчені у напрямку комп’ютерних наук намагалися знайти спосіб навчити машину розпізнавати візуальні дані ще 60 років тому. Сьогодні існує дуже багато інститутів, які знайшли спосіб впровадження комп’ютерного зору у медицині та продукції, але все ще існує чимало невирішених проблем.

Одне з питань полягає у тому, чи можливо маніпулювати тим, що бачить комп’ютер, використовуючи штучний інтелект? На даний момент машина здатна розпізнавати образи, але існує досить мало проектів, пов’язаних із модифікацією цих розпізнаних образів. Таким чином виникає задача, яку досить складно вирішити.

Обробка зображення та комп’ютерний зір довгий час існували окремо. Обробка зображення займалася безпосередньо додатковими ефектами та накладанням додаткових шарів на зображення, коли комп’ютерний зір відповідав за розпізнавання образів [2]. Ближче до наших часів вчені в інститутах почали комбінувати ці дві сфери в одну. Наприклад, нідерландський стартап створив алгоритм мімікування людської мови («підроблені новини»), де обличчя на відео оброблялось таким чином, що воно

наче промовляло те, що промовляє користувач на камеру [3]. Стартап з Каліфорнійського Університету створив алгоритм формування 3D-аватару, базуючись на 3D-моделі просканованої голови користувача [4]. Пізніше такі компанії-гіганти як Apple та Samsung підхопили цей тренд використання облич у якості аватарів. Активне дослідження та створення нових продуктів, зв'язаних з комбінуванням обробки зображення та комп'ютерного зору, доводить, що дана проблема є досить актуальною на сьогодні.

Знаходження способів маніпулювання візуальними даними, розпізнаних комп'ютерним зором, використовуючи інструменти обробки зображень може вирішити низку проблем:

- генерування нових візуальних даних, близьких до вже існуючих даних (наприклад, маючи лише одне зображення грабіжника, за допомогою інструментів комп'ютерного зору та обробки зображень можна згенерувати зображення того ж грабіжника у різних позах або з різними емоціями на обличчі);
- створення нових датасетів (англ. «datasets» - колекція великої кількості даних схожого типу [5]) зображень або відео без втручання людських ресурсів;
- тощо.

На сьогоднішній день цей напрям науки є досить непізнаним, частково через обмеженість інструментів для реалізації, а у випадку існування потрібних інструментів – їх обмежений доступ.

У даній роботі розглядається один з методів обробки розпізнаних комп'ютерним зором даних. Одним з головних аспектів є використання лише тих інструментів, які є у вільному доступі. За приклад було взято зміну емоцій на обличчі, маючи лише фотографію або відео. Більш точно це можна назвати «спотворенням розпізнаних образів на візуальних даних».

Загальний процес складається з двох етапів: розпізнавання самих об'єктів та його спотворення.

Існує досить різноманітний інструментарій для реалізації етапу розпізнавання образів, їх принцип у більшості випадків є однаковим. Створюється нейронна мережа, яку навчають розпізнавати орієнтири (англ. «landmarks») на об’єкті, який потрібно модифікувати, у даному випадку обличчі.

Для розпізнавання орієнтирів безпосередньо на обличчі можливо використати інструмент мапування (англ. «mapping»), при якому розпізнаються контури кожного об’єкту на зображенні [6]. Приклад мапування представлено на рисунку 1.1.

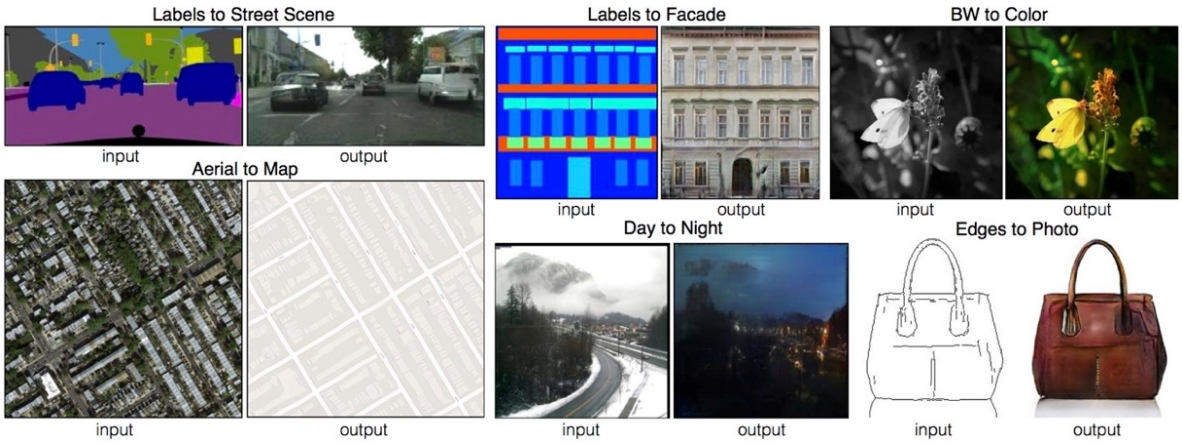


Рисунок 1.1 – Приклад мапування об'єктів на зображенні

Ще одним способом розпізнавання орієнтирів є знаходження ключових точок на об’єкті. Даний спосіб буде відрізнятися в залежності від виду об’єкту, який розпізнається [7]. Приклад розпізнавання ключових точок на обличчі зображено на рисунку 1.2.

Рисунок 1.2 – Приклад розпізнаних 194 ключових точок на обличчі

Другий етап спотворення зображення можливо реалузівати двома способами.

1) Заміна деяких частин на зображенні. Такий процес називається *заміною об'єктів* (англ. «Object Transplanting»).

Принцип заміни об’єктів у тому, щоб знайти релевантну частину об’єкта на одному зображені та замінити нею частину об’єкта на вхідному зображенні [8].

2) Обробка безпосередньо вхідного зображення, використовуючи алгоритми комп'ютерної графіки. Одними з таких алгоритмів є *алгоритми зміщення зображення* (англ. «Image Distortion»). Вони також поділяються на *хвильові алгоритми* (англ. «Wavy images algorithm»), *шарові алгоритми* (англ. «Sphere images algorithm») [9], також відомий як *фільтр «риб’яче око»* (англ. «Fisheye Filter») [10] і т.д. Для більш гладкого переходу також використовується алгоритм лінійної

інтерполяції [21]. Приклад роботи таких алгоритмів представлено на рисунках 1.3 та 1.4.

Рисунок 1.3 – Зображення з синус-хвильовим ефектом

Рисунок 1.4 – Сферовий ефект на зображенні

Мета вирішення даної проблеми полягає у тому, щоб знайти доступний спосіб генерування контенту з вже існуючого, використовуючи алгоритми штучного інтелекту та комп’ютерного зору, при цьому мінімально змінюючи ключові риси зображення.

1.1.1 Опис процесу діяльності

Проблему спотворення розпізнаних образів пропонується вирішити за допомогою нейронних мереж для знаходження орієнтирів та деяких інструментів обробки зображень. Спочатку нейронні мережі тренуються для розпізнавання орієнтирів на великій кількості зображень з обличчями у різному ракурсі та різних розмірів. Для більшої точності можливо реалізувати алгоритм мапування, який також базується на глибинному навчанні (англ. «Deep Learning» - наука про алгоритми, на базі структури та функціонування біологічного мозку. Такі алгоритми також називають штучними нейронними мережами [11]).

Після успішного розпізнавання орієнтирів на обличчі пропонується використати алгоритми обробки зображень, такі як хвильові та шарові алгоритми, а також лінійну інтерполяцію. Вони зміщують деякі частини на об’єкті і таким чином утворюють ефект «зміни емоції».

На рисунку 1.5 представлено діаграму діяльності загального алгоритму, який складається з двох етапів розпізнавання та спотворення. Схема структурна діаграми діяльності також представлена у графічному матеріалі.

Рисунок 1.5 – Схема структурна діяльності загального алгоритму

У підготовку візуальних даних входить [2]:

- позбавлення даних від шуму;
- підгонка до потрібного розміру;
- обрізання;
- перевірка формату;
- тощо.

Фіксація кута точки огляду відбувається шляхом поступової зміни кута зображення або відео.

1.1.2 Опис функціональної моделі

Так як дана реалізація є алгоритмом, який використовує інструменти штучного інтелекту та обробки зображень, у моделі є лише один актор – користувач. Задачею користувача є подання візуальних даних на вхід для подальшої обробки алгоритмом. Сам алгоритм діє у рамках програмного забезпечення, яке здатне приймати візуальні дані як через зовнішні девайси (веб-камера), так і через завантаження вже існуючих даних.

На рисунках 1.6, 1.7 та 1.8 представлено діаграму IDEF0, яка представляє функціональну модель алгоритму та програмного забезпечення, у яке його інтегровано.

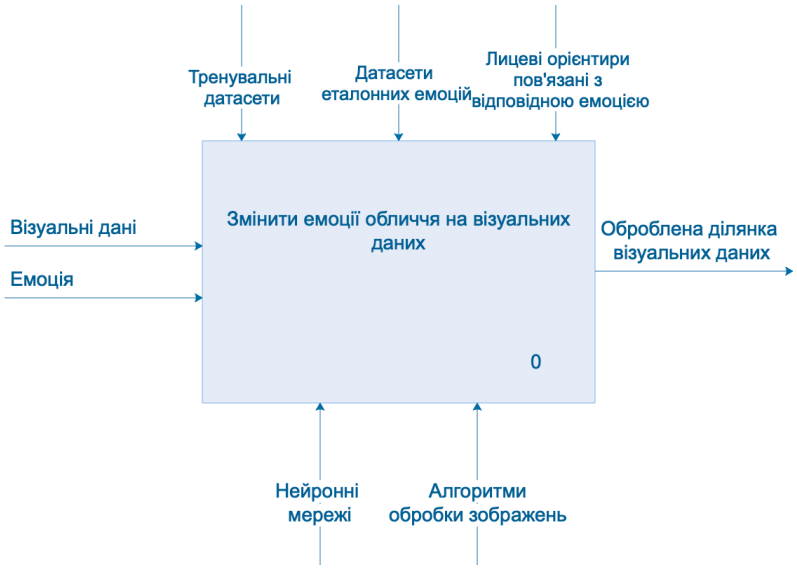


Рисунок 1.6 – Схема структурна контекстної моделі системи

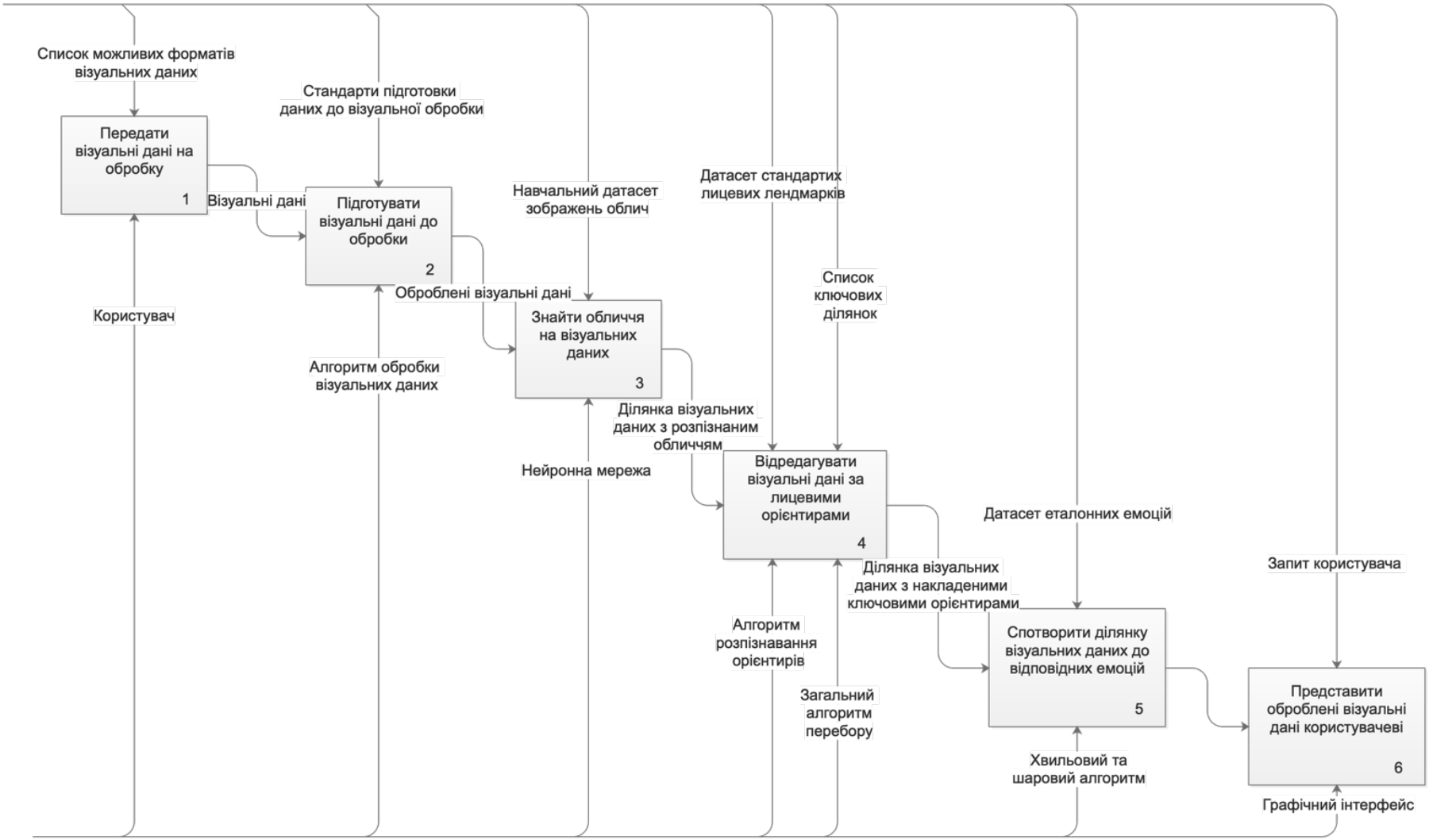


Рисунок 1.7 – Схема структурна функціональної моделі IDEF0

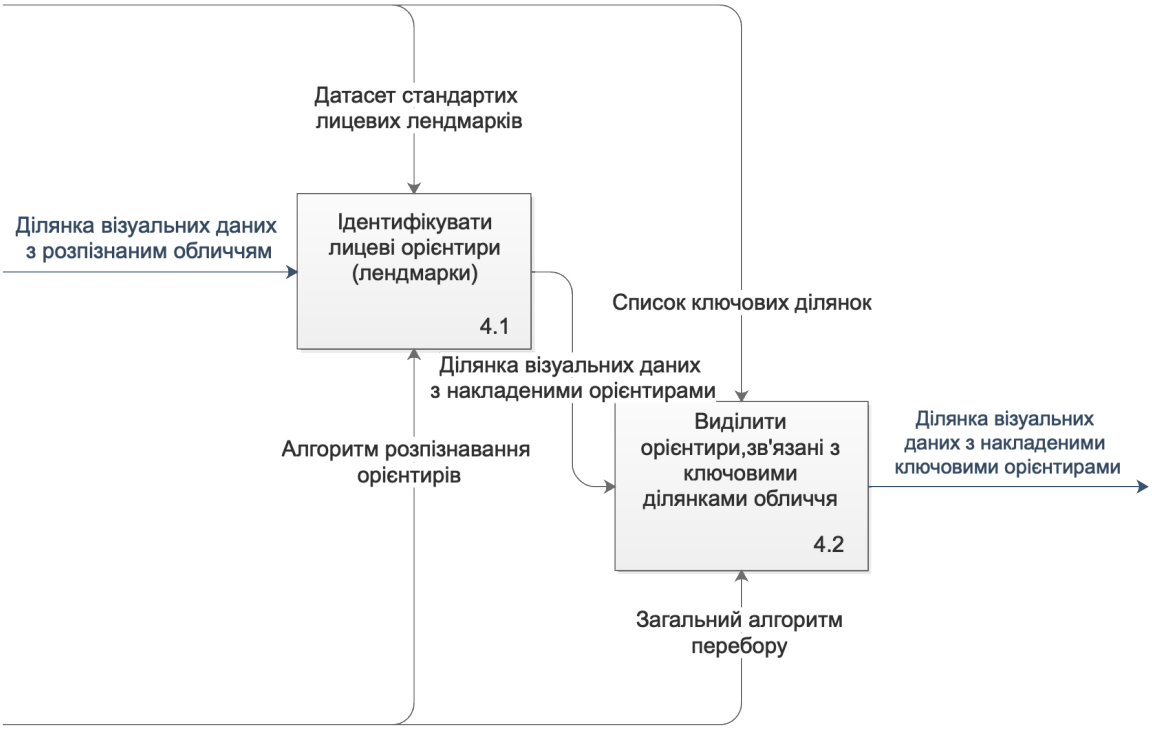


Рисунок 1.8 – Схема структурна декомпозиції процесу «Відредагувати візуальні дані за лицевими орієнтирами»

Після отримання візуальних даних від користувача система готує їх для подальшої обробки. У обробки входить підгонка та підрізання.

Після попередньої обробки система розпізнає обличчя на зображенні та підрізає візуальні дані під розпізнане обличчя.

Після вдалого розпізнавання обличчя обробляється для створення ефекту «посмішки».

Схеми структурні контекстної моделі системи та функціональної моделі IDEF0 також представлено у графічному матеріалі.

1.2 Огляд наявних аналогів

Під час пошуку схожих рішень було знайдено комерційний проект під назвою FaceApp. Проект є мобільним застосунком, створений російською компанією Wireless Lab, який використовує технологію нейронної мережі для автоматичного генерування реалістичних трансформацій на обличчях,

зображених на фотографіях. Застосунок здатний змінювати зображення обличчя так, щоб «примусити» його «посміхатися», виглядати молодше, старіше або навіть змінити стать. FaceApp було запущено у січні 2017 року.

На даний момент застосунок має понад 80 млн. активних користувачів та 21 фільтрів різного призначення, а також з можливістю отримання додаткових 7 фільтрів з пакетом, який потрібно купувати [12].

Рисунок 1.9 – Результат роботи застосунку «FaceApp»

Не дивлячись на те, що застосунок пропонує використання 21 фільтрів безкоштовно, реалізація більшості алгоритмів знаходиться у закритих репозиторіях, тобто код реалізації не є у відкритому доступі, що звужує коло розробників, які здатні використовувати ідею для своїх наступних продуктів.

Ще однією проблемою є той факт, що сам алгоритм обробки емоції на зображенні обличчя, базується на заміні деяких частин, використовуючи деталі з іншого зображення. Це означає, що для реплікації деякої емоції, нам потрібно використовувати зовнішні ресурси.

Іншим аналогом слугує частковий фільтр у «Snapchat», мобільному застосунку для соціальних мереж, створеному колишніми студентами Стенфордського Університету та компанією Snap Inc. Одними з ключовими

властивостями застосунку є те, що текстові та візуальні повідомлення доступні для перегляду лише лімітований час для їх одержувачів. Snapchat став особливо популярним після введення функції вставки фільтрів та стикерів, а також елементів доповненої реальності. У лютому 2018 року застосунок мав близько 187 млн користувачів щоденно. У травні 2019 року було введено новий фільтр з доповненою реальністю, який відтворює досить реалістичну посмішку на обличчі без використання додаткових візуальних даних для обробки.

Рисунок 1.10 – Результат роботи застосунку «Snapchat»

У якості ще одного аналога було знайдено проект «Deepfact», реалізованого нідерландським стартапом «3DUniversum». «Deepfact» - це проект мімікування зображенням людими на відео розмови, яку відтворює користувач на камеру [8]. Терміном «Deepfakes» називають підроблені візуальні дані, які були згенеровані та маніпульовані штучним інтелектом. Проект досяг достатньо великого успіху, а держава Королівства Нідерланди виступила офіційним спонсором.

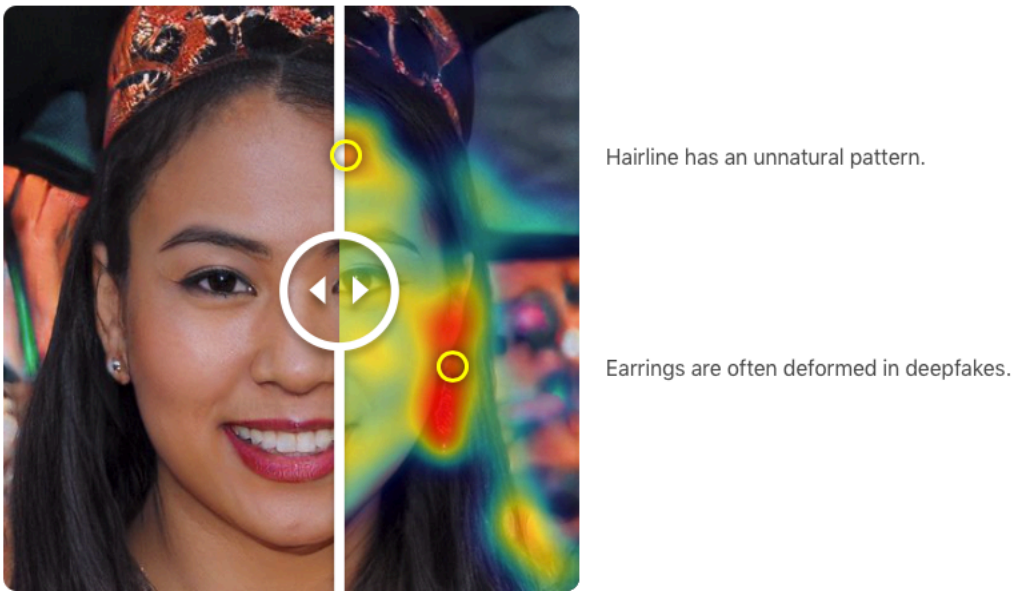


Рисунок 1.11 – Результат роботи «Deepfact»

Зазвичай згенеровані дані у проекті мають деякі дефекти, наприклад на фото, представленому на рисунку 1.11 не співпадають сережки та деякі структури на обличчі.

Як і з першою альтернативою, проект «Deepfacts» є закритим, це означає, що реалізація не є у вільному доступі.

1.3 Постановка задачі

1.3.1 Призначення розробки

Призначення розробки є створення системи, яка приймає на вхід візуальні дані різних форматів, розпізнає на них обличчя та змінює на ньому емоцію, при цьому не використовуючи елементів зображень з зовнішніх джерел.

1.3.2 Цілі та задачі розробки

Ціллю розробки є спростити обчислювальний процес генерування даних, використовуючи інструменти штучного інтелекту при цьому маючи лише візуальні дані одного типу.

Для досягнення цілі потрібно розв’язати наступні задачі:

- реалізувати процес генерування нових візуальних даних, який не потребує допоміжні інструменти;
- створити нові методи комбінування інструментів комп’ютерного зору та обробки зображень;
- полегшити маніпулювання візуальними даними;
- полегшити процес розробки систем, які одночасно використовують інструменти комп’ютерного зору та обробки зображень.

Висновок до розділу

Комп’ютерний зір та обробка зображень грають важливу роль у розвитку технологій. Здавна ці дві науки існували окремо, та існує чимало досліджень, які намагалися їх комбінувати. Було створено низку проектів, які давали можливість маніпулювати даними, наприклад «Deepfact», створений нідерландським стартапом, через який можна було маніпулювати образами на візуальних даних, використовуючи лише веб-камеру, через яку користувач подавав команди [8], або застосунок «Snapchat», який використовує доповнену реальність та комп’ютерний зір для створення унікальних фільтрів для фото та відео.

Знаходження способів маніпулювання візуальними даними, розпізнаних комп’ютерним зором, використовуючи інструменти обробки зображень може вирішити низку проблем:

- генерування нових візуальних даних, близьких до вже існуючих даних (наприклад, маючи лише одне зображення грабіжника, за допомогою інструментів комп’ютерного зору та обробки зображень можна згенерувати зображення того ж грабіжника у різних позах або з різними емоціями на обличчі);

- створення нових датасетів (англ. «datasets» - колекція великої кількості даних схожого типу [5]) зображень або відео без втручання людських ресурсів;
- тощо.

Однією з невирішених проблем також є знаходження способу відтворення схожих алгоритмів, маючи лише інструменти, які є у вільному доступі, а також мінімальну кількість візуальних даних на опрацювання.

У даній роботі процес знаходження рішення розбито на два етапи: розпізнавання образів на візуальних даних та їх спотворення. Такий спосіб також дозволить використовувати тільки оригінальні дані, які були подані на вхід, для видачі результату, замість знаходження та вилучення потрібних частин об’єкту з інших зображень або відео.

У якості прикладу було взято *зміну емоцій розпізнаного обличчя на візуальних даних*.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

У якості вхідних даних використовуються візуальні дані у вигляді зображення або відео. Вони подаються у двох режимах: у режимі реального часу та завантаження вже готового зображення або відео.

Візуальні дані конвертуються у двовимірний масив чисел, які вказують на колір кожної комірки (пікселя) на зображенні або відео. У випадку відео, дані конвертуються у багатовимірний масив.

2.2 Вихідні дані

У якості вихідних даних використовуються візуальні дані у вигляді ділянки зображення або відео.

Вхідні дані обробляються та обрізаються до ділянки, на якій зображено обличчя людини, після чого вихідні дані представляються багатовимірним масивом чисел, які позначають колір кожної комірки (пікселя), який є ділянкою візуальних даних із зміненою емоцією.

2.3 Структура масивів інформації

Для розробки програмного продукту використовується датасет AFLW [16], який використовується у нейронних мережах для наступних цілей:

- знаходження лицевих орієнтирів;
- розпізнавання обличь під різними кутами;
- оцінка позиції обличчя.

Датасет має 389 473 знайдених вручну лицевих орієнтирів з 21 точок на обличчі. Кількість відповідних орієнтирів на зображеннях представлено у таблиці 2.1.

Таблиця 2.1 – Структура датасету з лицевими орієнтирами

ID	Тип орієнтиру	Кількість знайдених
1	Ліва брова лівий кут	16,545
2	Ліва брова середина	20,624
3	Ліва брова правий кут	21,764
4	Права брова лівий кут	21,979
5	Права брова середина	20,790
6	Права брова правий кут	16,751
7	Ліве око лівий кут	19,461
8	Ліве око середина	21,439
9	Ліве око правий кут	18,183
10	Праве око лівий кут	17,877
11	Праве око середина	21,873
12	Праве око правий кут	19,569
13	Ліве вухо	10,885
14	Ніс зліва	18,217
15	Центр носу	25,993
16	Ніс праворуч	18,647
17	Праве вухо	11,684
18	Лівий кут рота	20,482
29	Центр рота	25,448
20	Правий кут рота	21,262
21	Центр підборіддя	24,641
Всього		**389,473**

На рисунку 2.1 представлено розташування відповідних лицевих орієнтирів на обличчі.

Рисунок 2.1 – Розташування лицевих орієнтирів на обличчі

На рисунку 2.2 представлено структуру повного датасету з зображеннями.

Name	Size
aflw-db.tar.gz	8.9 MB
aflw-db.tar.gz.md5	49 bytes
aflw-facedbsql-src.tar.gz	2.4 MB
aflw-facedbsql-src.tar.gz.md5	60 bytes
aflw-facedetector.tar.gz	142.5 KB
aflw-facedetector.tar.gz.md5	59 bytes
aflw-gui-bin-win32.zip	8.5 MB
aflw-gui-bin-win32.zip.md5	57 bytes
aflw-gui-src.tar.gz	14.5 MB
aflw-gui-src.tar.gz.md5	54 bytes
aflw-images-0.tar.gz	3.2 GB
aflw-images-0.tar.gz.md5	55 bytes
aflw-images-2.tar.gz	3.0 GB
aflw-images-2.tar.gz.md5	55 bytes
aflw-images-3.tar.gz	2.6 GB
aflw-images-3.tar.gz.md5	55 bytes
aflw-matlab.tar.gz	14.6 KB
aflw-matlab.tar.gz.md5	53 bytes
aflw_example.py	5.1 KB
hashes.md5	381 bytes

Рисунок 2.2 – Структура датасету

Висновок до розділу

Дана програмна розробка приймає на вхід візуальні дані, які можуть бути у вигляді як зображення, так і відео. Також передбачено два способи подання візуальних даних: у режимі реального часу та завантаження даних у систему.

Вихідними даними є ті ж самі візуальні дані, але в обробленому вигляді, зазвичай обрізані до зображення самого обличчя.

Для реалізації необхідно створити нейронну мережу, яка розпізнає лицеві орієнтири на обличчі, тому було обрано датасет AFLW [16] для тренування мережі, який є основним масивом інформації.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

Задача заміни емоції на зображенні полягає у знаходженні розв'язків низки підзадач.

Перша задача полягає у розпізнаванні потрібного об'єкту на зображенні, у даному випадку обличчі. Вона розв'язується за допомогою нейронних мереж, які тренуються на датасеті вже готових даних з відповідями. Через подібне навчання формується функція, яка здатна класифікувати дані схожого типу у майбутньому.

Після успішного розпізнавання обличчя на візуальних даних, розв'язується задача розміщення лицевих орієнтирів на розпізнаних даних. У результаті отримуються дані про орієнтири та їх координати. Проблема обробки зображення для зміни емоції на розпізнаному обличчі розглядається тільки після того, як розпізнано позиції кожної точки лицевих орієнтирів.

3.2 Математична постановка задачі

Задано $a: X \to Y$, де X є множиною зображень, а Y – множина класифікаторів, які можуть приймати значення 1 або 0, що відповідно означає «обличчя» та «не обличчя». Існує деяке відображення, значення якого відоме тільки на об'єктах кінцевої навчальної вибірки $X^i = \{(x_1, y_1), \dots, (x_i, y_i)\}$. На даних результуючому відображення будується функція y , яка може приймати значення 1 або 0. Отже функція $y(M)$ є функцією класифікації об'єкту на зображенні і визначає, чи є об'єкт обличчям, де M – будь-який об'єкт на зображенні.

Для $\forall\, y(M) = 1$ створюється матриця S, яка позначає значення кожного пікселю на зображенні у кожній комірці.

$$S = \begin{bmatrix} s_{11} & \cdots & s_{1n} \\ \vdots & \ddots & \vdots \\ s_{m1} & \cdots & s_{mn} \end{bmatrix},$$

$s_{ij} = \overline{1,256}, \quad m,n -$ довжина та ширина зображення

Нехай $A -$ множина лицевих орієнтирів з q елементів. Функція y^* ставить у відповідність кожен лицевий орієнтир з його позицією на зображенні і отримує нову множину $A^* = \left\{\left(a_1, \left(s_{i_1 j_1}, s_{k_1 l_1}\right)\right), \ldots, \left(a_q, \left(s_{i_q j_q}, s_{k_q l_q}\right)\right)\right\}$.

Для $\forall S^* \exists$ функція $f = f(A, S^*) = R$, де $R -$ матриця однакової розмірності з матрицею A з деякою кількістю змінених значень комірок. R визначає оброблене зображення із зміненою емоцією на розпізнаному обличчі.

Задача полягає у знаходженні функцій y, y^* та f.

3.3 Обґрунтування методу розв'язання

Розв'язання задачі відбувається через розв'язання низки підзадач. Для розпізнавання обличчя та лицевих орієнтирів на зображенні найефективнішим способом є використання окремих нейронних мереж. Результатом роботи двох функцій, знайдених після реалізації нейронних мереж, є дані про кожний лицевий орієнтир, поставлений у відповідність з його координатами на візуальних даних, які можуть бути як відео, так і зображенням.

Маючи координати ключових лицевих орієнтирів, розв'язується наступна задача спотворення візуальних даних для досягнення мети, у даному випадку зміни зображення емоції. Для цього знаходиться функція спотворення, яка накладає сферовий та хвильовий ефекти обробки зображення.

3.4 Опис методів розв'язання

Є декілька способів створення та навчання нейронних мереж для розпізнавання обличчя та лицевих орієнтирів, які, у більшості випадків,

відрізняються між собою функцією активації та способом «підгонки» значень ваг до потрібних [17].

У даній роботі пропонується використати мультикласову логістичну регресію у загальному алгоритмі розпізнавання обличчя [18].

Припустимо, що ми маємо множину X вхідних даних, з яких потрібно класифікувати вектор вхідних даних $x^{(i)}$ у значення $y^{(i)} \in Y$. Нехай маємо m таких пар: $\{\left(x^{(i)}, y^{(i)}\right), i = 1, \dots, m\}$. Задача полягає у тому, щоб знайти таку функцію $h: X \to Y$, що $h(x)$ мала достатньо «високу» точність прогнозування відповідного значення y. Така функція h називається гіпотезою [19]. Ймовірність даних визначається за допомогою $p(\vec{y}|X;\theta)$, де θ – константи (ваги нейронних мереж). При представленні функції у вигляді функції, яка залежить від значення θ, то отримуємо так звану функцію правдоподібності

$$L(\theta) = L(\theta; X, \vec{y}) = p(\vec{y}|X;\theta).$$

У логістичній регресії гіпотеза $h_\theta(x)$ здатна приймати будь-яке значення менше 1 та більше 0. Для цього використовується логістична або сигмоїдна функція $h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$, де $g(z) = \frac{1}{1+e^{-z}}$. Графік функції представлено на рисунку 3.1.

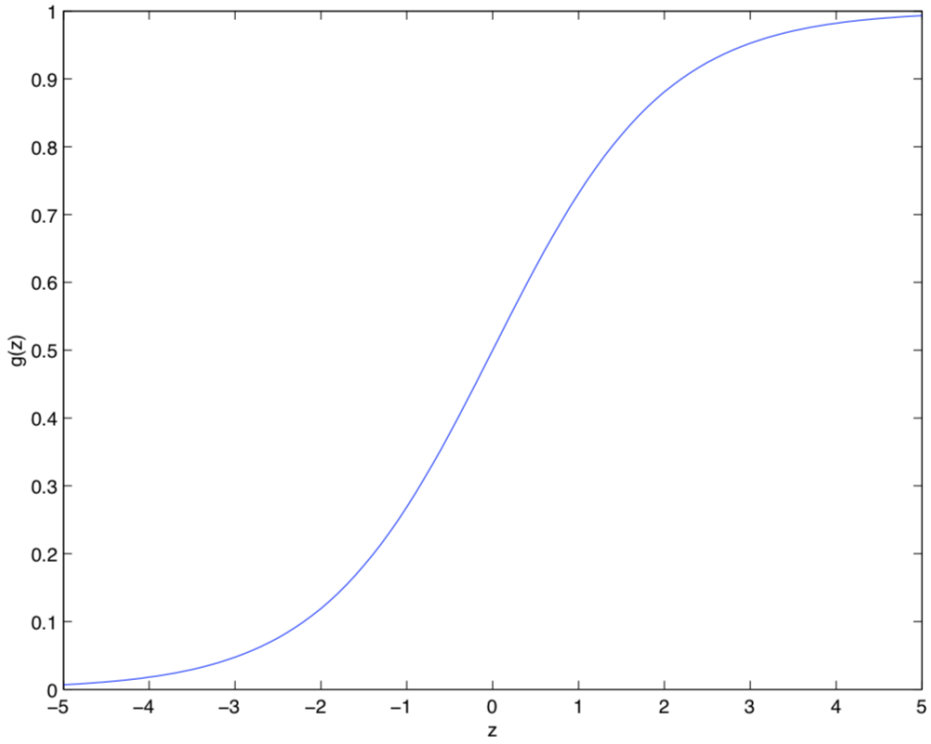


Рисунок 3.1 – Функція $g(z)$

$g(z)$ прямує до 1 при $z \to \infty$, $g(z)$ прямує до 0 при $z \to -\infty$. Так як $g(z)$ обмежена одиницею та нулем, то і $h(x)$ теж.

Припустимо, що наразі у нас два класи: $y = 0$, якщо зображення не є обличчям, $y = 1$, якщо навпаки.

Тоді виходить наступне припущення:

$P(y = 1|\, x; \theta) = h_\theta(x)$,

$P(y = 0|\, x; \theta) = 1 - h_\theta(x)$, що в свою чергу може бути представлене наступним чином:

$$p(y \mid x; \theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y}.$$

Припускаючи, що m пар даних було згенеровано незалежно один від одного, ми можемо записати функцію правдоподібності наступним чином:

$$L(\theta) = L(\theta; X, \vec{y}) = p(\vec{y} \mid X; \theta)$$

$$= \prod_{i=1}^{m} p\left(y^{(i)} | x^{(i)}; \theta\right) = \prod_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right)\right)^{y^{(i)}} \left(1 - h_\theta\left(x^{(i)}\right)\right)^{1-y^{(i)}}.$$

Максимізуємо функцію правдоподібності. Для цього зведемо її до логарифму для простішої обробки.

$$l(\theta) = \log L(\theta) = \sum_{i=1}^{m} y^{(i)} \log h\left(x^{(i)}\right) + (1 - y^{(i)}) \log\left(1 - h\left(x^{(i)}\right)\right).$$

Використаємо градієнтний спуск з використанням диференціювання: $\theta := \theta + \alpha \nabla_\theta l(\theta)$.

На прикладі однієї навчальної пари диференціал буде виглядати наступним чином:

$$\frac{d}{d\theta_j} l(\theta) = \left(y \frac{1}{g(\theta^T x)} - (1 - y) \frac{1}{1 - g(\theta^T x)}\right) \frac{d}{d\theta_j} g(\theta^T x)$$

$$= \left(y \frac{1}{g(\theta^T x)} - (1 - y) \frac{1}{1 - g(\theta^T x)}\right) g(\theta^T x)\left(1 - g(\theta^T x)\right) \frac{d}{d\theta_j} \theta^T x$$

$$= (y\left(1 - g(\theta^T x)\right) - (1 - y) g(\theta^T x) x_j = \left(y - h_\theta(x)\right) x_j.$$

Таким чином отримуємо правило стохастичного градієнтного спуску:

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta\left(x^{(i)}\right)\right) x_j{}^{(i)}.$$

Загальний алгоритм складається з двох етапів: локалізація обличчя та безпосередньо його розпізнавання. Всі два процеси виконуються за допомогою знаходження спільних рис із вже існуючим наобором зображень облич у базі нейронної мережі. Зображення обличчя моделюється як двовимірний масив чисел, які репрезентують відповідний піксель.

$X = \{x_i, i \in S\}$, де S – двовимірний простір.

Іноді усе зображення презентується у вигляді одного вектора пікселів $X = [x_1, x_2, \dots, x_N]^T$, де N – загальна кількість пікселей у зображенні.

Результатом порівняння ключових рис є вектор $f(X) = [f_1(X), f_2(X), \dots, f_M(X)]^T$, де $f_1(x), f_2(x) \dots$ є лінійними та нелінійними функціями та $M < N$ [18].

Алгоритм спотворення зображення працює за принципом спотворення «liquify» та лінійної інтерполяції. У комп'ютерній графіці район обробки

будемо називати «областю інтересу», у даному випадку районом інтересу є область роту, тобто масив пікселів $X = \{x_i, i \in S\}$. Для спотворення координати переводяться з декартової системи у полярну — (r, α), після чого відстань r конвертується у нове значення $r' = c\sqrt{r}$, де $c -$ константа.

Для більш гладких країв використовується лінійна інтерполяція, під час якої визначається коефіцієнт спотворення. Якщо піксель знаходиться більше до границі, то коефіцієнт спотворення менше, якщо піксель знаходиться ближче до центру – навпаки. Для цього знаходиться $\frac{r}{radius}$, де $radius$ — радіус області інтересу, а r — координата поточного пікселю. Таким чином отримуємо остаточну формулу спотворення зображення:

$$r' = \frac{r}{radius} r + (1 - \frac{r}{radius}) c\sqrt{r}.$$

Висновок до розділу

Отже, головною задачею є знайти функції розпізнавання та локалізації лицевих орієнтирів, а також функції обробки зображення для спотворення розпізнаного образу обличчя.

Зазвичай, будь-які візуальні дані можна представити у вигляді набору двовимірних масивів, а у деяких випадках і єдиного вектору. Перший етап розпізнавання пропонується розв’язати за допомогою нейронних мереж, а саме мультикласової логістичної регресії у загальному алгоритмі розпізнавання обличчя з використанням логістичної або сигмоїдної функції $h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$, де $g(z) = \frac{1}{1+e^{-z}}$ та градієнтного спуску, отриманого з максимізації функції правдоподібності

$$\theta_j := \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j{}^{(i)}$$

[18].

Загальний алгоритм складається з двох етапів: локалізація обличчя та безпосередньо його розпізнавання. Всі два процеси виконуються за

допомогою знаходження спільних рис із вже існуючим наобором зображень облич у базі нейронної мережі. Зображення обличчя моделюється як двовимірний масив чисел, які репрезентують відповідний піксель.

$X = \{x_i, i \in S\}$, де S – двовимірний простір.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

У даній роботі використовуються такі інструменти як Eclipse IDE, бібліотеки та фреймворки OpenCV та TensorFlow, інструменти для роботи з файлами формату .csv, а також мова програмування Swift та середовище Xcode із зберігання даних у SQLite.

Мови програмування Python та C++ використовуються для роботи з TensorFlow, фреймворку для роботи з нейронними мережами, та OpenCV, найбільшою бібліотекою для створення алгоритмів комп'ютерного зору.

Мова програмування Python дає змогу легко та швидко запрограмувати алгоритм обробки та розпізнавання образів, що робить її ідеальною для прототипування та випробування різних способів імплементації нейронних мереж.

За допомогою мови програмування C++ можливо створити ті ж самі алгоритми, але з більшою ефективністю роботи через те, що її код потрібно компілювати після кожного внесення змін. Робота з мовою C++ також відрізняється від роботи з Python тим, що на написання коду та тестування потрібно витратити набагато більше часу через більш складний синтаксис. Бібліотека OpenCV підтримує обидві мови програмування, що робить цю пару найбільш вигідною для роботи над складними системами комп'ютерного зору.

Для взаємодії із системою створюється мобільний застосунок мовою Swift та середовищем програмування Xcode. Для зберігання отриманих результатів використовується Core Data – засіб для зберігання об'єктів у базі даних SQLite.

4.2 Вимоги до технічного забезпечення

4.2.1 Загальні вимоги

Алгоритми глибинного навчання, які грають ключову роль у напрямку комп'ютерного зору, потребують досить потужної техніки для їх стабільної роботи.

До загальних вимог входить використання потужних GPU, такі як Nvidia, та CPU з не менше 4 ядрами, RAM обсягу не менше 8гб, а також можливість навчання нейронної мережі на постійній основі як мінімум 48 годин без переривання.

Для взаємодії із системою та її діагностики також необхідно мати планшет iPad не пізніше 2017 року з платформою iOS версії не раніше 12.2, а також комп'ютер, який працює на операційній системі MacOS версії не раніше 10.14.5 з середовищем програмування Xcode як мінімум 9 версії.

4.3 Архітектура програмного забезпечення

4.3.1 Діаграма класів

На рисунку 4.1 представлено діаграму класів системи. Схема структурна класів системи також представлена у графічному матеріалі.

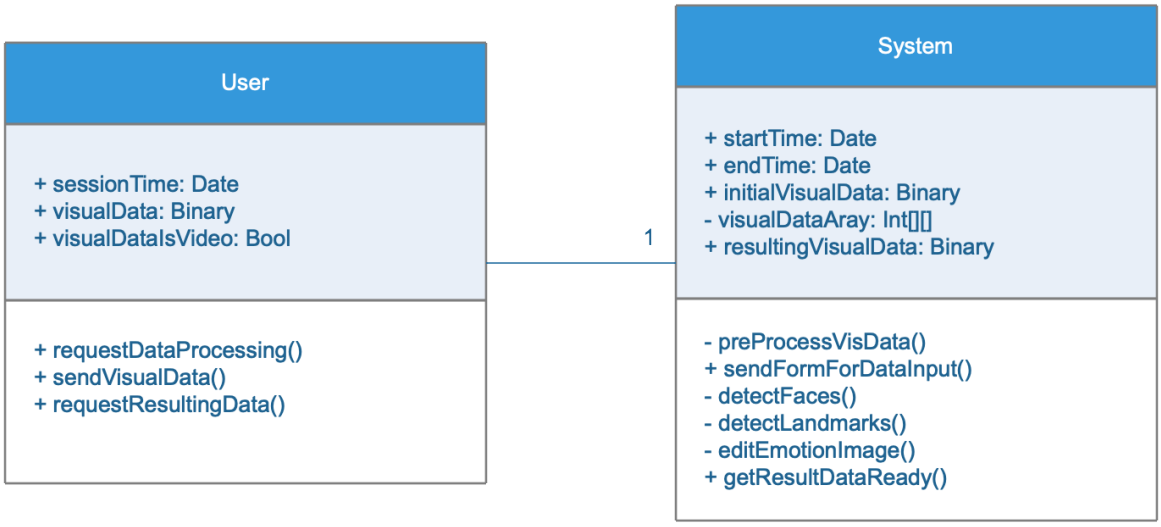


Рисунок 4.1 – Схема структурна класів програмного забезпечення

На діаграмі класів представлено два основні класи: Користувач та Система. У кожного класу є свої атрибути та операції.

Клас Користувач має наступні атрибути: час запиту на обробку візуальних даних, самі візуальні дані, а також булевий атрибут, який позначає, чи є візуальні дані відео чи зображення. Серед операцій є наступні: відправлення запиту на нову обробку даних, відправлення візуальних даних на обробку, відправлення запиту на перегляд результуючих даних.

Існує також допоміжний клас для збережених результатів, який має лише два атрибути: дата створення результату та посилання на сам результат.

4.3.2 Діаграма послідовності

На рисунку 4.2 представлено діаграму послідовності системи.

Рисунок 4.2 – Схема структурна послідовності системи

На діаграмі зображено послідовність дій та передачі повідомлень між користувачем та системою. Система починає свою роботу безпосередньо тільки після запиту користувача на нову обробку візуальних даних. Для цього користувач відсилає окремий запит з типом візуальних даних, які він хоче обробити.

Після отримання нових візуальних даних, система обробляє їх: готує до процесу розпізнавання обличчя та лицевих орієнтирів, розпізнає їх та приводить дані до вигляду, зрозумілого для користувача (наприклад, конвертує вигляд багатовимірної матриці у відповідне зображення). Після успішної обробки система повідомляє користувача про завершення роботи.

Додатковим кроком поза діаграми є пропозиція зберегти результат у базі даних.

Схема структурна послідовності системи також представлена у графічному матеріалі.

4.3.3 Діаграма компонентів

На рисунку 4.3 зображено діаграму компонентів.

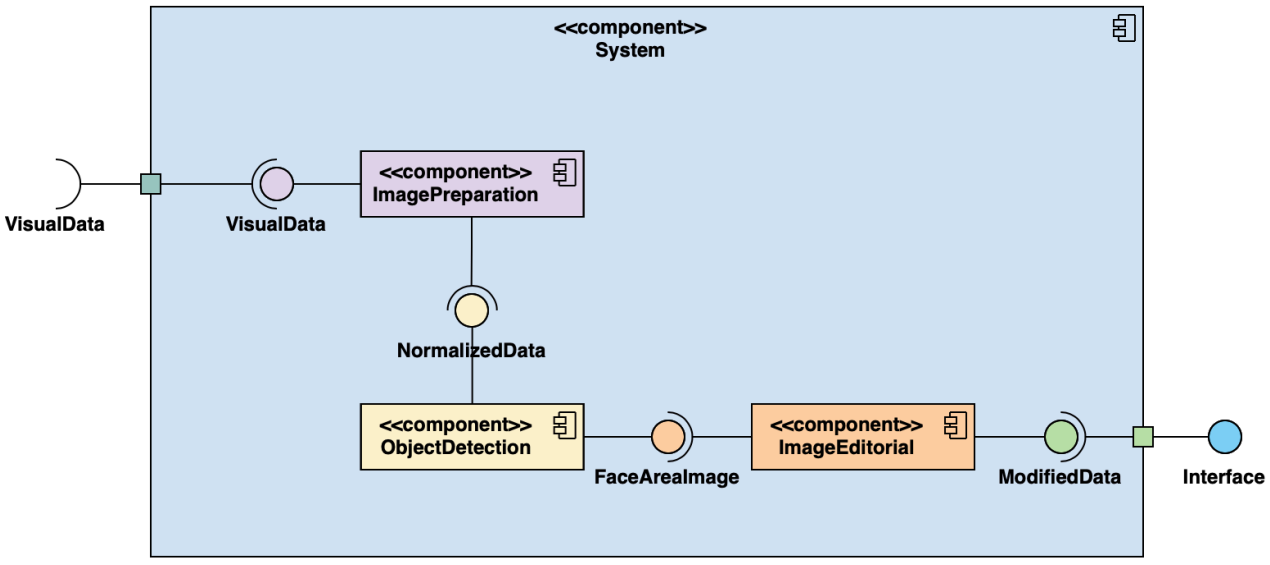


Рисунок 4.3 – Схема структурна компонентів системи

На діаграмі зображено компоненти безпосередньо у системі. Для початку її роботи обов'язково потрібні візуальні дані, подані на вхід користувачем.

У системі компоненти відповідають за наступні процеси: загальна обробка та нормалізація вхідних візуальних даних, розпізнавання обличчя та лицевих орієнтирів та модифікація емоцій на зображенні обличчя.

4.3.4 Специфікація функцій

Функції класів програмного забезпечення наведені у таблиці 4.1.

Таблиця 4.1 – Функції класів програмного забезпечення

Назва	**Примітка**
Клас: User – відповідає за дії користувача	
public requestDataProcessing()	Створює запит на нову обробку візуальних даних та відправляє його у систему
public sendVisualData()	Приймає нові візуальні дані та відправляє їх у систему
public requestResultingData()	Створює запит на перегляд результуючих візуальних даних та відправляє його у систему
Клас: System – головний клас, який відповідає за роботу системи	
private preprocessVisData()	Готує отримані візуальні дані до обробки
public sendFormForDataInput()	Відправляє користувачеві форму для завантаження візуальних даних
private detectFaces()	Використовується для розпізнавання обличчя на візуальних даних
private detectLandmarks()	Використовується для розпізнавання лицевих орієнтирів на розпізнаному обличчі

Продовження таблиці 4.1

private editEmotionImage()	Використовується для обробки зображення емоції на розпізнаному обличчі
public getResultDataReady()	Повертає візуальним даним старий формат для представлення результату на екрані користувача
public saveData()	Допоміжний клас для збереження результату у базі даних

Висновок до розділу

Дана система розробляється за допомогою таких засобів розробки як мови програмування Python та Swift. Вони використовуються для прототипування алгоритмів та більш ефективного та швидкого навчання нейронних мереж.

Архітектура програмного забезпечення складається з двох класів, Користувач та Система, а також трьох компонентів у самій системі. Задачею класу Користувач в більшості є відправлення запитів та отримання результату, коли Система відповідає за решту частину процесу розпізнавання образів та обробки візуальних даних.

Загалом у системі присутні наступні функції:

- створення запиту на нову обробку візуальних даних та відправлення його у систему;
- прийняття нових візуальних даних та відправлення їх на обробку;
- створення запиту на перегляд результуючих візуальних даних;
- підготовка отриманих візуальних даних до обробки;

- відправлення користувачеві форму для завантаження візуальних даних;
- розпізнавання обличчя на візуальних даних;
- розпізнавання лицевих орієнтирів на розпізнаному обличчі;
- обробка зображення емоції на розпізнаному обличчі;
- повернення візуальним даним старого формату для представлення результату на екрані користувача;
- збереження результату у базі даних.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Рисунок 5.1 – Результат роботи системи

5.2 Випробування програмного продукту

У процесі тестування було перевірено функціонування ряду комплексу задач, з них об’єктами випробування є функції вводу та виводу візуальних даних, функції перевірок на правильність вхідних та вихідних візуальних даних. Нижче приведено перелік основних функціонувань, поданих на перевірку.

Таблиця 5.1 – Відправлення запиту на нову обробку

Мета тесту:	**Перевірка функції «Відправлення запиту на нову обробку»**
Початковий стан КЗ	Відкрито інтерфейс «Home» програми
Вхідні данні:	Формат «Video»
Схема проведення тесту:	Відмітити формат «Video»
	Натиснути кнопку «Send request»
Очікуваний результат:	Відкрита форма для завантаження відео
Стан КЗ після проведення випробувань:	Відкрито інтерфейс «Home» з формою для завантаження відео

Таблиця 5.2 – Перевірка завантаження даних у режимі реального часу

Мета тесту:	**Перевірка функції «Завантаження даних у режимі реального часу»**
Початковий стан КЗ	Відкрито інтерфейс «Home» з формою для завантаження відео
Вхідні данні:	Відео з веб-камери
Схема проведення тесту:	Обрати «Take a video» Зняти на відео обличчя з різними рухами Натиснути на «Send»

Продовження таблиці 5.2

Очікуваний результат:	Повідомлення про успішне завантаження даних на обробку
Стан КЗ після проведення випробувань:	Відкрито інтерфейс з прогресуючим індикатором виконання

Таблиця 5.3 – Перевірка завантаження даних з девайсу

Мета тесту:	**Перевірка функції «Завантаження даних з девайсу»**
Початковий стан КЗ	Відкрито інтерфейс «Home» з формою для завантаження відео
Вхідні данні:	Відео з девайсу
Схема проведення тесту:	Обрати «Load a video» Обрати відео з диску Натиснути кнопку «Send request»
Очікуваний результат:	Повідомлення про успішне завантаження даних на обробку
Стан КЗ після проведення випробувань:	Відкрито інтерфейс з прогресуючим індикатором виконання

Таблиця 5.4 – Перевірка виходу із системи під час її роботи

Мета тесту:	**Перевірка функції «Зберігання даних»**
Початковий стан КЗ	Відкрито інтерфейс з прогресуючим індикатором виконання
Вхідні данні:	-
Схема проведення тесту:	Натиснути «Close»
	Відкрити програму
Очікуваний результат:	Повідомлення про незавершену обробку та пропозиція на продовження процесу
Стан КЗ після проведення випробувань:	Відкрите вікно пропозиції продовжити процес

Таблиця 5.5 – Перегляд обробленого результату форматі відео

Мета тесту:	**Перевірка функції «Зберігання даних»**
Початковий стан КЗ	Відкрито інтерфейс з повідомленням про успішну обробку
Вхідні данні:	-
Схема проведення тесту:	Натиснути «Show results»
Очікуваний результат:	Інтерфейс з представленою обробленою ділянкою відео
Стан КЗ після проведення випробувань:	Відкрите вікно з програвачем відео

5.2.1 Мета випробувань

Метою випробувань є перевірка на коректну роботу основних функцій системи, зв’язаних з користувачем.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;
– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробувань

Випробування показали позитивний показник окрім нереалізованого функціоналу завантаження відео з девайсу.

Висновок до розділу

У процесі тестування було перевірено функціонування ряду комплексу задач, з них об’єктами випробування є функції вводу та виводу візуальних даних, функції перевірок на правильність вхідних та вихідних візуальних даних.

У даній роботі було розроблено наступний функціонал: обробка зображення у режимі реального часу, зберігання даних, та обробка завантаженого фото. Розроблені функції працювали коректно за виключенням обробки зображення у режимі реального часу. За час, виділений розробці програмного продукту вдалося розробити лише часткову модифікацію відео, а саме обробка окремого фрейму.

ЗАГАЛЬНІ ВИСНОВКИ

Стрімкий розвиток штучного інтелекту спричинив народженню таких напрямків як комп'ютерний зір, а з існуванням комп'ютерів завжди мало сенс розвивати і комп'ютерну графіку. Але не завжди комп'ютерний зір та комп'ютерну графіку намагалися поєднати разом. Та при правильному використанні обох інструментів можливо створити неймовірні програмні продукти, які можуть не раз стати у нагоді.

У даній роботі було досліджено метод відтворення системи обробки зображень розпізнаних об'єктів на прикладі облич людей без використання зовнішніх візуальних даних для обробки вхідних даних, а також із використанням лише інструментів розробки, які перебували у вільному доступі.

Знаходження способів маніпулювання візуальними даними, розпізнаних комп'ютерним зором, використовуючи інструменти обробки зображень розв'язує низку проблем, у тому числі автоматизації генерування даних за допомогою штучного інтелекту та покращення ефективності обробки зображення.

У даній роботі процес знаходження рішення було розбито на два етапи: розпізнавання образів на візуальних даних та їх спотворення. Через таке нестандартне рішення було усунуто випадок використання зовнішніх візуальних даних для обробки вхідних візуальних даних.

Акцент було зроблено на *зміні емоцій розпізнаного обличчя на візуальних даних*.

Для реалізації системи було створено нейронну мережу на основі датасету AFLW для тренування, алгоритми локалізації обличчя та безпосередньо його розпізнавання. Всі два алгоритми використовували знаходження спільних рис із вже існуючим наобором зображень облич у базі нейронної мережі. Візуальні дані системи було спроектовано у вигляді двовимірного масиву

$X = \{x_i, i \in S\}$, де S – двовимірний простір.

Було використано такі засоби розробки як мови програмування Python та Swift. Вони використовуються для прототипування алгоритмів та більш ефективного та швидкого навчання нейронних мереж.

Загалом було розроблено наступний функціонал: обробка зображення у режимі реального часу, зберігання даних, та обробка завантаженого фото. Розроблені функції працювали коректно за виключенням обробки зображення у режимі реального часу. За час, виділений розробці програмного продукту вдалося розробити лише часткову модифікацію відео, а саме обробка окремого фрейму.

ПЕРЕЛІК ПОСИЛАНЬ

1. http://paulbourke.net/dome/fish2/
2. https://www.vpro.nl/programmas/tegenlicht/kijk/afleveringen/2018-2019/deep-fake-news.html
3. https://www.pinscreen.com
4. https://hayo.io/computer-vision/
5. https://github.com/affinelayer/pix2pix-tensorflow
6. https://www.vpro.nl/programmas/tegenlicht/kijk/afleveringen/2018-2019/deep-fake-news.html
7. https://arxiv.org/pdf/1705.04932.pdf
8. https://deepfact.3duniversum.com
9. https://arxiv.org/pdf/1705.04932.pdf
10. http://www.filtermeister.com/docs/da.pdf chapter 6.3 Wavy Images, page 56
11. https://machinelearningmastery.com/what-is-deep-learning/
12. https://faceapp.com/
13. https://medium.com/@pallawi.ds/difference-between-image-processing-computer-vision-and-artificial-intelligence-af670d65055d
14. https://www.codeproject.com/Articles/3419/Image-Processing-for-Dummies-with-C-and-GDI-Part-5
15. https://en.wikipedia.org/wiki/Data_set
16. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.384.2988&rep=rep1&type=pdf
17. An Introduction to Neural Networks by Kevin Gurney
18. https://core.ac.uk/download/pdf/26858086.pdf
19. http://cs229.stanford.edu/notes/cs229-notes1.pdf
20. https://arxiv.org/pdf/1812.04948.pdf
21. *https://www.encyclopediaofmath.org/index.php/Linear_interpolation*

Додаток А

Тексти програмного коду

Система обробки зображень облич людей за допомогою алгоритмів комп'ютерного зору

DVD-R

(Вид носія даних)

968 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2019 року

Алгоритм розпізнавання лицевих орієнтирів

```
import AVFoundation
import UIKit
import Vision

class FaceDetectionViewController: UIViewController {
    var sequenceHandler = VNSequenceRequestHandler()

    @IBOutlet var faceView: FaceView!
    @IBOutlet var laserView: LaserView!
    @IBOutlet var faceLaserLabel: UILabel!

    let session = AVCaptureSession()
    var previewLayer: AVCaptureVideoPreviewLayer!

    let dataOutputQueue = DispatchQueue(
        label: "video data queue",
        qos: .userInitiated,
        attributes: [],
        autoreleaseFrequency: .workItem)

    var faceViewHidden = false

    var maxX: CGFloat = 0.0
    var midY: CGFloat = 0.0
    var maxY: CGFloat = 0.0

    override func viewDidLoad() {
        super.viewDidLoad()
        configureCaptureSession()

        laserView.isHidden = true

        maxX = view.bounds.maxX
        midY = view.bounds.midY
        maxY = view.bounds.maxY

        session.startRunning()
    }
}

// MARK: - Gesture methods
```

```
extension FaceDetectionViewController {
    @IBAction func handleTap(_ sender: UITapGestureRecognizer) {
        faceView.isHidden.toggle()
        laserView.isHidden.toggle()
        faceViewHidden = faceView.isHidden

        if faceViewHidden {
            faceLaserLabel.text = "Lasers"
        } else {
            faceLaserLabel.text = "Face"
        }
    }
}

// MARK: - Video Processing methods

extension FaceDetectionViewController {
    func configureCaptureSession() {
        // Define the capture device we want to use
        guard let camera = AVCaptureDevice.default(.builtInWideAngleCamera,
                                                   for: .video,
                                                   position: .front) else {
                                                    fatalError("No front video camera available")
        }

        // Connect the camera to the capture session input
        do {
            let cameraInput = try AVCaptureDeviceInput(device: camera)
            session.addInput(cameraInput)
        } catch {
            fatalError(error.localizedDescription)
        }

        // Create the video data output
        let videoOutput = AVCaptureVideoDataOutput()
        videoOutput.setSampleBufferDelegate(self, queue: dataOutputQueue)
        videoOutput.videoSettings = [kCVPixelBufferPixelFormatTypeKey as
String: kCVPixelFormatType_32BGRA]

        // Add the video output to the capture session
        session.addOutput(videoOutput)
```

```
        let videoConnection = videoOutput.connection(with: .video)
        videoConnection?.videoOrientation = .portrait

        // Configure the preview layer
        previewLayer = AVCaptureVideoPreviewLayer(session: session)
        previewLayer.videoGravity = .resizeAspectFill
        previewLayer.frame = view.bounds
        view.layer.insertSublayer(previewLayer, at: 0)
    }
}

// MARK: - AVCaptureVideoDataOutputSampleBufferDelegate methods

extension FaceDetectionViewController:
AVCaptureVideoDataOutputSampleBufferDelegate {
    func captureOutput(_ output: AVCaptureOutput, didOutput sampleBuffer:
CMSampleBuffer, from connection: AVCaptureConnection) {
        // 1
        guard let imageBuffer = CMSampleBufferGetImageBuffer(sampleBuffer)
else {
            return
        }

        // 2
        let detectFaceRequest =
VNDetectFaceLandmarksRequest(completionHandler: detectedFace)

        // 3
        do {
            try sequenceHandler.perform(
                [detectFaceRequest],
                on: imageBuffer,
                orientation: .leftMirrored)
        } catch {
            print(error.localizedDescription)
        }
    }
}

extension FaceDetectionViewController {
    func convert(rect: CGRect) -> CGRect {
        // 1
```

```
        let origin = previewLayer.layerPointConverted(fromCaptureDevicePoint:
rect.origin)

        // 2
        let size = previewLayer.layerPointConverted(fromCaptureDevicePoint:
rect.size.cgPoint)

        // 3
        return CGRect(origin: origin, size: size.cgSize)
    }

    // 1
    func landmark(point: CGPoint, to rect: CGRect) -> CGPoint {
        // 2
        let absolute = point.absolutePoint(in: rect)

        // 3
        let converted =
previewLayer.layerPointConverted(fromCaptureDevicePoint: absolute)

        // 4
        return converted
    }

    func landmark(points: [CGPoint]?, to rect: CGRect) -> [CGPoint]? {
        guard let points = points else {
            return nil
        }

        return points.compactMap { landmark(point: $0, to: rect) }
    }

    func updateFaceView(for result: VNFaceObservation) {
        defer {
            DispatchQueue.main.async {
                self.faceView.setNeedsDisplay()
            }
        }

        let box = result.boundingBox
        faceView.boundingBox = convert(rect: box)

        guard let landmarks = result.landmarks else {
```

```
        return
    }

    if let leftEye = landmark(
        points: landmarks.leftEye?.normalizedPoints,
        to: result.boundingBox) {
        faceView.leftEye = leftEye
    }

    if let rightEye = landmark(
        points: landmarks.rightEye?.normalizedPoints,
        to: result.boundingBox) {
        faceView.rightEye = rightEye
    }

    if let leftEyebrow = landmark(
        points: landmarks.leftEyebrow?.normalizedPoints,
        to: result.boundingBox) {
        faceView.leftEyebrow = leftEyebrow
    }

    if let rightEyebrow = landmark(
        points: landmarks.rightEyebrow?.normalizedPoints,
        to: result.boundingBox) {
        faceView.rightEyebrow = rightEyebrow
    }

    if let nose = landmark(
        points: landmarks.nose?.normalizedPoints,
        to: result.boundingBox) {
        faceView.nose = nose
    }

    if let outerLips = landmark(
        points: landmarks.outerLips?.normalizedPoints,
        to: result.boundingBox) {
        faceView.outerLips = outerLips
    }

    if let innerLips = landmark(
        points: landmarks.innerLips?.normalizedPoints,
        to: result.boundingBox) {
        faceView.innerLips = innerLips
```

```
        }

        if let faceContour = landmark(
            points: landmarks.faceContour?.normalizedPoints,
            to: result.boundingBox) {
            faceView.faceContour = faceContour
        }
    }

    // 1
    func updateLaserView(for result: VNFaceObservation) {
        // 2
        laserView.clear()

        // 3
        let yaw = result.yaw ?? 0.0

        // 4
        if yaw == 0.0 {
            return
        }

        // 5
        var origins: [CGPoint] = []

        // 6
        if let point = result.landmarks?.leftPupil?.normalizedPoints.first {
            let origin = landmark(point: point, to: result.boundingBox)
            origins.append(origin)
        }

        // 7
        if let point = result.landmarks?.rightPupil?.normalizedPoints.first {
            let origin = landmark(point: point, to: result.boundingBox)
            origins.append(origin)
        }

        // 1
        let avgY = origins.map { $0.y }.reduce(0.0, +) / CGFloat(origins.count)

        // 2
        let focusY = (avgY < midY) ? 0.75 * maxY : 0.25 * maxY
```

```
        // 3
        let focusX = (yaw.doubleValue < 0.0) ? -100.0 : maxX + 100.0

        // 4
        let focus = CGPoint(x: focusX, y: focusY)

        // 5
        for origin in origins {
            let laser = Laser(origin: origin, focus: focus)
            laserView.add(laser: laser)
        }

        // 6
        DispatchQueue.main.async {
            self.laserView.setNeedsDisplay()
        }
    }

    func detectedFace(request: VNRequest, error: Error?) {
        // 1
        guard
            let results = request.results as? [VNFaceObservation],
            let result = results.first
            else {
                // 2
                faceView.clear()
                return
        }

        if faceViewHidden {
            updateLaserView(for: result)
        } else {
            updateFaceView(for: result)
        }
    }
}
```

Алгоритм спотворення зображення

```
var radius = Math.floor(size / 2);

var centerX = Math.floor(width / 2);
var centerY = Math.floor(height / 2);
```

```
var posY, posX, position;


func liquifySimple(canvasId, originalImageData) {
   "use strict";
   var x, y, width, height, size, radius, centerX, centerY, sourcePosition,
destPosition;

   var transformedImageData = createCompatibleImageData(canvasId,
originalImageData);
   var originalPixels = originalImageData.data;
   var transformedPixels = transformedImageData.data;
   var r, alpha, angle;
   var newX, newY;
   var degrees;
   var radiusSquared = radius * radius;
   var c;

   // Iterate over the square interest region
   for (y = -radius; y < radius; ++y) {
      for (x = -radius; x < radius; ++x) {
         // Translate the x, y coordinates to the image center
         posY = y + centerY;
         posX = x + centerX;

         // Use the pixel position formula
         position = posY * width + posX;
         position *= 4;
      }
   }

   width = originalImageData.width;
   height = originalImageData.height;

   centerX = Math.floor(width / 2);
   centerY = Math.floor(height / 2);
   size = width < height ? width : height;
   radius = Math.floor(size / 2);

   copyImageData(originalPixels, transformedPixels, width, height);

   // Iterate over the interest square region
   for (y = -radius; y < radius; ++y) {
```

```
            for (x = -radius; x < radius; ++x) {
               // Check if the pixel is inside the effect circle
               if (x * x + y * y <= radius * radius) {
                  // Get the pixel array position
                  destPosition = (y + centerY) * width + x + centerX;
                  destPosition *= 4;

                  // Transform the pixel Cartesian coordinates (x, y) to polar coordinates (r, alpha)
                  r = Math.sqrt(x * x + y * y);
                  alpha = Math.atan2(y, x);

                  // Remember that the angle alpha is in radians, transform it to degrees
                  degrees = (alpha * 180.0) / Math.PI;

                  c = 12.5;
                  r = c * Math.sqrt(r);

                  // Transform back from polar coordinates to Cartesian
                  alpha = (degrees * Math.PI) / 180.0;
                  newY = Math.floor(r * Math.sin(alpha));
                  newX = Math.floor(r * Math.cos(alpha));

                  // Get the new pixel location
                  sourcePosition = (newY + centerY) * width + newX + centerX;
                  sourcePosition *= 4;

                  transformedPixels[destPosition + 0] = originalPixels[sourcePosition + 0];
                  transformedPixels[destPosition + 1] = originalPixels[sourcePosition + 1];
                  transformedPixels[destPosition + 2] = originalPixels[sourcePosition + 2];
                  transformedPixels[destPosition + 3] = originalPixels[sourcePosition + 3];
               }
            }
         }

         drawPixels(canvasId, transformedImageData);
      }
```

Код інтерфейсу користувача

```
<?xml version="1.0" encoding="UTF-8"?>
<document
type="com.apple.InterfaceBuilder3.CocoaTouch.Storyboard.XIB"
version="3.0" toolsVersion="14460.31" targetRuntime="iOS.CocoaTouch"
propertyAccessControl="none" useAutolayout="YES"
useTraitCollections="YES" useSafeAreas="YES" colorMatched="YES"
initialViewController="Aar-lP-KaY">
    <device id="retina4_7" orientation="portrait">
      <adaptation id="fullscreen"/>
    </device>
    <dependencies>
      <deployment identifier="iOS"/>
      <plugIn identifier="com.apple.InterfaceBuilder.IBCocoaTouchPlugin"
version="14460.20"/>
      <capability name="Named colors" minToolsVersion="9.0"/>
      <capability name="Safe area layout guides" minToolsVersion="9.0"/>
      <capability name="documents saved in the Xcode 8 format"
minToolsVersion="8.0"/>
    </dependencies>
    <scenes>
      <!--Face Detection View Controller-->
      <scene sceneID="tR7-cA-c2e">
        <objects>
          <viewController id="Aar-lP-KaY"
customClass="FaceDetectionViewController" customModule="FaceLasers"
customModuleProvider="target" sceneMemberID="viewController">
            <view key="view" contentMode="scaleToFill" id="U9Y-eo-
qQq">
              <rect key="frame" x="0.0" y="0.0" width="375"
height="667"/>
              <autoresizingMask key="autoresizingMask"
widthSizable="YES" heightSizable="YES"/>
              <subviews>
                <view contentMode="scaleToFill"
translatesAutoresizingMaskIntoConstraints="NO" id="9SK-4K-rvp"
customClass="FaceView" customModule="FaceLasers"
customModuleProvider="target">
                  <rect key="frame" x="0.0" y="0.0" width="375"
height="667"/>
```

```
                    <color key="backgroundColor" white="0.0" alpha="0.0"
colorSpace="custom"
customColorSpace="genericGamma22GrayColorSpace"/>
                </view>
                <view                    contentMode="scaleToFill"
translatesAutoresizingMaskIntoConstraints="NO"          id="Czg-41-0ht"
customClass="LaserView"                customModule="FaceLasers"
customModuleProvider="target">
                    <rect key="frame" x="0.0" y="0.0" width="375"
height="667"/>
                    <color key="backgroundColor" white="0.0" alpha="0.0"
colorSpace="custom"
customColorSpace="genericGamma22GrayColorSpace"/>
                    <gestureRecognizers/>
                </view>
                <visualEffectView opaque="NO" clipsSubviews="YES"
contentMode="scaleToFill" translatesAutoresizingMaskIntoConstraints="NO"
id="a1G-aU-0il">
                    <rect key="frame" x="142" y="626" width="91"
height="41"/>
                    <view key="contentView" opaque="NO"
clipsSubviews="YES" multipleTouchEnabled="YES" contentMode="center"
insetsLayoutMarginsFromSafeArea="NO" id="Tlw-5G-CBH">
                        <rect key="frame" x="0.0" y="0.0" width="91"
height="41"/>
                        <autoresizingMask          key="autoresizingMask"
widthSizable="YES" heightSizable="YES"/>
                        <subviews>
                            <label                              opaque="NO"
userInteractionEnabled="NO"                          contentMode="left"
horizontalHuggingPriority="251" verticalHuggingPriority="251" text="Face"
textAlignment="center"                lineBreakMode="tailTruncation"
baselineAdjustment="alignBaselines"          adjustsFontSizeToFit="NO"
translatesAutoresizingMaskIntoConstraints="NO" id="oCN-6N-WUE">
                                <rect key="frame" x="20" y="10" width="51"
height="21"/>
                                <constraints>
                                    <constraint          firstAttribute="height"
constant="21" id="a6M-8q-J3F"/>
                                    <constraint          firstAttribute="width"
relation="greaterThanOrEqual" constant="51" id="neo-xZ-2Kg"/>
                                </constraints>
```

```
                                <fontDescription key="fontDescription"
type="system" pointSize="17"/>
                                <color key="textColor" name="rw-light"/>
                                <nil key="highlightedColor"/>
                            </label>
                        </subviews>
                        <userDefinedRuntimeAttributes>
                            <userDefinedRuntimeAttribute type="number"
keyPath="cornerRadius">
                                <real key="value" value="0.0"/>
                            </userDefinedRuntimeAttribute>
                        </userDefinedRuntimeAttributes>
                    </view>
                    <constraints>
                        <constraint firstAttribute="width" constant="91"
id="2hu-rB-kvE"/>
                        <constraint firstAttribute="height" constant="41"
id="AmK-xR-oLl"/>
                        <constraint firstItem="oCN-6N-WUE"
firstAttribute="centerY" secondItem="a1G-aU-0il" secondAttribute="centerY"
id="F7Y-nX-FgB"/>
                        <constraint firstItem="oCN-6N-WUE"
firstAttribute="centerX" secondItem="a1G-aU-0il" secondAttribute="centerX"
id="YK5-Aa-Puw"/>
                    </constraints>
                    <blurEffect style="light"/>
                    <userDefinedRuntimeAttributes>
                        <userDefinedRuntimeAttribute type="number"
keyPath="cornerRadius">
                            <real key="value" value="13"/>
                        </userDefinedRuntimeAttribute>
                    </userDefinedRuntimeAttributes>
                </visualEffectView>
            </subviews>
            <color key="backgroundColor" white="0.0" alpha="1"
colorSpace="custom"
customColorSpace="genericGamma22GrayColorSpace"/>
            <gestureRecognizers/>
            <constraints>
                <constraint firstAttribute="top" secondItem="Czg-41-0ht"
secondAttribute="top" id="6P8-QH-Bp7"/>
```

```
                    <constraint firstItem="9SK-4K-rvp"
firstAttribute="centerX" secondItem="Czg-41-0ht" secondAttribute="centerX"
id="8Hs-HN-vkI"/>
                    <constraint firstItem="Czg-41-0ht"
firstAttribute="leading" secondItem="U9Y-eo-qQq" secondAttribute="leading"
id="Adn-Pn-zBD"/>
                    <constraint firstItem="9SK-4K-rvp"
firstAttribute="height" secondItem="Czg-41-0ht" secondAttribute="height"
id="DhD-FR-azP"/>
                    <constraint firstAttribute="bottom" secondItem="Czg-41-
0ht" secondAttribute="bottom" id="Kdp-kw-APR"/>
                    <constraint firstItem="5E3-3m-hrQ"
firstAttribute="bottom" secondItem="a1G-aU-0il" secondAttribute="bottom"
id="V5A-MX-mnW"/>
                    <constraint firstAttribute="trailing" secondItem="Czg-41-
0ht" secondAttribute="trailing" id="m0J-Yv-NzC"/>
                    <constraint firstItem="9SK-4K-rvp" firstAttribute="width"
secondItem="Czg-41-0ht" secondAttribute="width" id="mzT-Za-lqb"/>
                    <constraint firstItem="a1G-aU-0il"
firstAttribute="centerX" secondItem="U9Y-eo-qQq"
secondAttribute="centerXWithinMargins" id="uj3-C2-6Bl"/>
                    <constraint firstItem="9SK-4K-rvp"
firstAttribute="centerY" secondItem="Czg-41-0ht" secondAttribute="centerY"
id="xTj-gd-VcT"/>
                </constraints>
                <viewLayoutGuide key="safeArea" id="5E3-3m-hrQ"/>
                <connections>
                    <outletCollection property="gestureRecognizers"
destination="Gax-pf-er4" appends="YES" id="Y9h-Ee-WNx"/>
                </connections>
            </view>
            <connections>
                <outlet property="faceLaserLabel" destination="oCN-6N-
WUE" id="dZC-EB-n8C"/>
                <outlet property="faceView" destination="9SK-4K-rvp"
id="huL-hY-fKf"/>
                <outlet property="laserView" destination="Czg-41-0ht"
id="6HC-Ra-hzM"/>
            </connections>
        </viewController>
        <placeholder placeholderIdentifier="IBFirstResponder" id="lhd-
pc-gnY" userLabel="First Responder" sceneMemberID="firstResponder"/>
        <tapGestureRecognizer id="Gax-pf-er4">
```

```
                    <connections>
                        <action selector="handleTap:" destination="Aar-lP-KaY"
id="rLU-al-k4b"/>
                    </connections>
                </tapGestureRecognizer>
            </objects>
            <point key="canvasLocation" x="-234.40000000000001"
y="160.56971514242881"/>
        </scene>
    </scenes>
    <resources>
        <namedColor name="rw-light">
            <color red="0.94901960784313721"
green="0.9647058823529419" blue="0.98039215686274506" alpha="1"
colorSpace="custom" customColorSpace="sRGB"/>
        </namedColor>
    </resources>
</document>
```

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації та управління

УЗГОДЖЕНО	**ЗАТВЕРДЖУЮ**
Керівник проекту	**В.о. завідувача кафедри**
_____ *Т. В. Ковалюк*	_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)	(підпис) (ініціали, прізвище)
“15” квітня 2019 р.	“15” квітня 2019 р.

СИСТЕМА ОБРОБКИ ЗОБРАЖЕНЬ ОБЛИЧ ЛЮДЕЙ ЗА ДОПОМОГОЮ АЛГОРИТМІВ КОМП’ЮТЕРНОГО ЗОРУ

ТЕХНІЧНЕ ЗАВДАННЯ

Шифр *ДП ІС-5218.1181-с.ТЗ*

на 10 сторінках

Київ – 2019 року

ЗМІСТ

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1. Повне найменування системи та її умовне позначення

Повне найменування системи: «*Система обробки зображень облич людей за допомогою алгоритмів комп'ютерного зору*».

Коротке найменування системи: «*Система обробки зображень облич людей*».

1.2. Найменування організації-замовника та організацій-учасників робіт

Замовником проекту є підприємство ФОП Крик Андріан Іванович. Адреса замовника: м. Вишневе, вулиця Машинобудівників, буд. 15а.

Розробником є студентка групи ІС-52 кафедри Автоматизованих систем обробки інформації та управління Національного технічного університету України "Київський політехнічний інститут ім. І. Сікорського" Нго Май Фионг.

1.3. Перелік документів, на підставі яких створюється система

При розробці системи і створення проектно-експлуатаційної документації Виконавець повинен керуватися вимогами наступних нормативних документів:

- ДСТУ 19.201-78. Технічне завдання. Вимоги до змісту і оформлення;
- ДСТУ 34.601-90. Комплекс стандартів на автоматизовані системи. Автоматизовані системи. Стадії створення;
- ДСТУ 34.201-89. Інформаційні технології. Комплекс стандартів на автоматизовані системи. Види, комплексність і позначення документів при створенні автоматизованих систем.

1.4. Планові терміни початку і закінчення роботи зі створення системи

Плановий строк початку роботи по створенню системи 15 квітня 2019 року. Плановий строк кінця роботи 19 травня 2019 року.

2 ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ

2.1. Призначення системи

Призначення розробки є створення системи, яка приймає на вхід візуальні дані різних форматів, розпізнає на них обличчя та змінює на ньому емоцію, при цьому не використовуючи елементів зображень з зовнішніх джерел.

Розробка системи може вирішити низку проблем:

- генерування нових візуальних даних, близьких до вже існуючих даних (наприклад, маючи лише одне зображення грабіжника, за допомогою інструментів комп'ютерного зору та обробки зображень можна згенерувати зображення того ж грабіжника у різних позах або з різними емоціями на обличчі);
- створення нових датасетів зображень або відео без втручання людських ресурсів.

2.2. Цілі створення системи

Ціллю розробки є спростити обчислювальний процес генерування даних, використовуючи інструменти штучного інтелекту при цьому маючи лише візуальні дані одного типу.

Для досягнення цілі потрібно розв'язати наступні задачі:

- реалізувати процес генерування нових візуальних даних, який не потребує допоміжні інструменти;
- створити нові методи комбінування інструментів комп'ютерного зору та обробки зображень;
- полегшити маніпулювання візуальними даними;
- полегшити процес розробки систем, які одночасно використовують інструменти комп'ютерного зору та обробки зображень.

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

До процесів автоматизації належать:

- Розпізнавання облич на зображенні;
- знаходження лицевих орієнтирів на розпізнаному обличчі;
- модифікування зображення.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Вимоги до функціональних характеристик

Задача обробки зображень облич людей повинна надавати відповідний зручний інтерфейс для роботи з поданими та отриманими даними.

Задача обробки зображень облич людей надає зручний інтерфейс для завантажування даних.

Задача обробки зображень облич людей вимагає утворення відповідних звітів для перегляду результату.

Задача обробки зображень облич людей вимагає надання коректних даних і вигляді зображення з обличчям.

4.2. Вимоги до надійності

Система повинна зберігати працездатність та забезпечувати відновлення своїх функцій при виникненні наступних позаштатних ситуацій:

- при збоях в системі або проблемах у електропостачанні;
- при введенні некоректних даних, які непередбачувані програмним продуктом;
- при незапланованих перезапусках системи.

4.3. Умови експлуатації

Для нормальної експлуантації розроблюваної системи має бути забезпечене постійне підключення до мережі Інтернет та безпроблемне живлення для девайсу.

Технічне обслуговування включає в себе тестування використовуваного обладнання, а також серверів та пристроїв живлення.

Розміщення обладнання, технічних засобів повинно відповідати вимогам техніки безпеки.

Всі користувачі системи повинні дотримуватися правил експлуатації електронної обчислювальної техніки.

4.4. Вимоги до складу і параметрів технічних засобів

Для правильної роботи серверної частини програмного забезпечення необхідно використовувати мінімальний склад технічних засобів:

- GPU, такі як Nvidia;
- CPU з не менше 4 ядрами;
- RAM обсягу не менше 8гб;
- операційна система з ядром UNIX.

Для використання мобільного застосування необхідно мати пристрій на ОС iOS з версією не нижче 12.2 та RAM як мінімум 2гб.

5 СТАДІЇ І ЕТАПИ РОЗРОБКИ

У таблиці 5.1 наведено календарний план робіт та терміни їх виконання.

Таблиця 5.1 – Календарний план виконання робіт

№ з/п	Назва етапів виконання дипломного проекту	Строк виконання
1.	Вивчення рекомендованої літератури	03.04.2019
2.	Аналіз існуючих методів розв’язання задачі	05.04.2019
3.	Постановка та формалізація задачі	05.04.2019
4.	Розробка інформаційного забезпечення	15.04.2019
5.	Алгоритмізація задачі	18.04.2019
6.	Обґрунтування використовуваних технічних засобів	21.04.2019
7.	Розробка програмного забезпечення	05.05.2019
8.	Налагодження програми	10.05.2019
9.	Виконання графічних документів	13.05.2019
10.	Оформлення пояснювальної записки	20.05.2019
11.	Подання ДП на попередній захист	30.05.2019
12.	Подання ДП на основний захист	01.06.2019
13.	Подання ДП рецензенту	18.06.2019

5 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ СИСТЕМИ

5.1. Види випробувань

Види випробувань узгоджуються з замовником до проведення випробувань. Здача - прийом робіт виконується поетапно на комп’ютерах замовника у відповідності з робочою програмою та календарним планом.

Всі програмні продукти, що створюються в рамках даної системи передаються замовнику як у вигляді готових модулів, так і у вигляді вихідних кодів, представлених в електронній формі.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації та управління

УЗГОДЖЕНО

Керівник проекту

_____ *Ковалюк Т. В.*
(підпис) (ініціали, прізвище)

“16” травня 2019 р.

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)

“17” травня 2019 р.

СИСТЕМА ОБРОБКИ ЗОБРАЖЕНЬ ОБЛИЧ ЛЮДЕЙ ЗА ДОПОМОГОЮ АЛГОРИТМІВ КОМП'ЮТЕРНОГО ЗОРУ

ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

Шифр *ДП ІС-5218.1181-с.ПМВ*

на 10 сторінках

Київ – 2019 року

ЗМІСТ

1 ОБ'ЄКТ ВИПРОБУВАННЯ

1.1 Найменування програми

Повне найменування програми: «*Система обробки зображень облич людей за допомогою алгоритмів комп'ютерного зору*».

Коротке найменування програми: «*Система обробки зображень облич людей*».

1.2 Область застосування

Програма застосовується для автоматичного генерування нових, розпізнавання та обробки існуючих візуальних даних.

1.3 Умовне позначення програми

Надалі умовно програма позначатиметься як СОЗОЛ.

2 МЕТА ВИПРОБУВАНЬ

Випробування програми проводяться з метою перевірки на коректність роботи вказаного у вимогах функціоналу.

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Вимоги до функціональних характеристик

Задача обробки зображень облич людей повинна надавати відповідний зручний інтерфейс для роботи з поданими та отриманими даними.

Задача обробки зображень облич людей надає зручний інтерфейс для завантажування даних.

Задача обробки зображень облич людей вимагає утворення відповідних звітів для перегляду результату.

Задача обробки зображень облич людей вимагає надання коректних даних і вигляді зображення з обличчям.

3.1.1 Вимоги до складу виконуваних функцій

Для правильної роботи серверної частини програмного забезпечення необхідно використовувати мінімальний склад технічних засобів:

- GPU, такі як Nvidia;
- CPU з не менше 4 ядрами;
- RAM обсягу не менше 8гб;
- операційна система з ядром UNIX.

Для використання мобільного застосування необхідно мати пристрій на ОС iOS з версією не нижче 12.2 та RAM як мінімум 2гб.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Вимоги до програмної документації базуються на основі наступних документів:

- ГОСТ РД 19.101–77. Єдина система програмної документації. Види програм та програмних документів;
- ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5 СКЛАД І ПОРЯДОК ВИПРОБУВАНЬ

У процесі тестування було перевірено функціонування ряду комплексу задач, з них об’єктами випробування є функції вводу та виводу візуальних даних, функції перевірок на правильність вхідних та вихідних візуальних даних. Нижче приведено перелік основних функціонувань, поданих на перевірку.

Таблиця 5.1 – Відправлення запиту на нову обробку

Мета тесту:	**Перевірка функції «Відправлення запиту на нову обробку»**
Початковий стан КЗ	Відкрито інтерфейс «Home» програми
Вхідні данні:	Формат «Video»
Схема проведення тесту:	Відмітити формат «Video»
	Натиснути кнопку «Send request»
Очікуваний результат:	Відкрита форма для завантаження відео
Стан КЗ після проведення випробувань:	Відкрито інтерфейс «Home» з формою для завантаження відео

Таблиця 5.2 – Перевірка завантаження даних у режимі реального часу

Мета тесту:	**Перевірка функції «Завантаження даних у режимі реального часу»**

Мета тесту:	**Перевірка функції «Завантаження даних у режимі реального часу»**
Початковий стан КЗ	Відкрито інтерфейс «Home» з формою для завантаження відео
Вхідні данні:	Відео з веб-камери
Схема проведення тесту:	Обрати «Take a video»
	Зняти на відео обличчя з різними рухами
	Натиснути на «Send»
Очікуваний результат:	Повідомлення про успішне завантаження даних на обробку
Стан КЗ після проведення випробувань:	Відкрито інтерфейс з прогресуючим індикатором виконання

Таблиця 5.3 – Перевірка завантаження даних з девайсу

Мета тесту:	**Перевірка функції «Завантаження даних з девайсу»**
Початковий стан КЗ	Відкрито інтерфейс «Home» з формою для завантаження відео
Вхідні данні:	Відео з девайсу
Схема проведення тесту:	Обрати «Load a video»

Мета тесту:	Перевірка функції «Завантаження даних з девайсу»
	Обрати відео з диску
	Натиснути кнопку «Send request»
Очікуваний результат:	Повідомлення про успішне завантаження даних на обробку
Стан КЗ після проведення випробувань:	Відкрито інтерфейс з прогресуючим індикатором виконання

Таблиця 5.4 – Перевірка виходу із системи під час її роботи

Мета тесту:	Перевірка функції «Зберігання даних»
Початковий стан КЗ	Відкрито інтерфейс з прогресуючим індикатором виконання
Вхідні данні:	-
Схема проведення тесту:	Натиснути «Close»
	Відкрити програму
Очікуваний результат:	Повідомлення про незавершену обробку та пропозиція на продовження процесу
Стан КЗ після проведення випробувань:	Відкрите вікно пропозиції продовжити процес

Таблиця 5.5 – Перегляд обробленого результату форматі відео

Мета тесту:	Перевірка функції «Зберігання даних»

Мета тесту:	**Перевірка функції «Зберігання даних»**
Початковий стан КЗ	Відкрито інтерфейс з повідомленням про успішну обробку
Вхідні данні:	-
Схема проведення тесту:	Натиснути «Show results»
Очікуваний результат:	Інтерфейс з представленою обробленою ділянкою відео
Стан КЗ після проведення випробувань:	Відкрите вікно з програвачем відео

6 МЕТОДИ ВИПРОБУВАНЬ

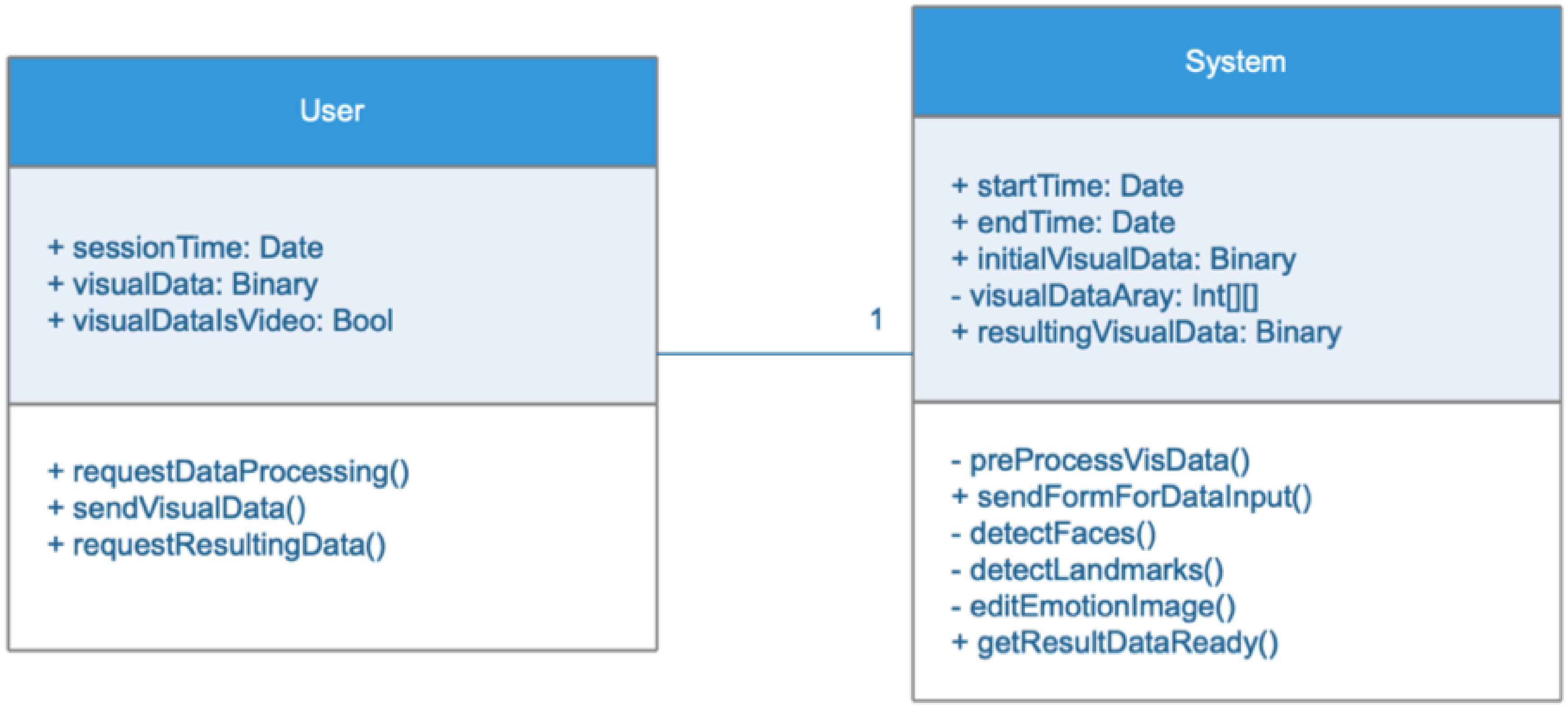
Випробування проводяться через перевірку роботи окремих частин коду програми, а також за допомогою тестування «чорної скриньки».

Графічний матеріал
до дипломного проекту

на тему: «Система обробки зображень облич людей за допомогою алгоритмів комп'ютерного зору»

Київ – 2019 року

ДП ІС-5218.1181-с.СКЗ



ДП ІС-5218.1181-с.СКЗ

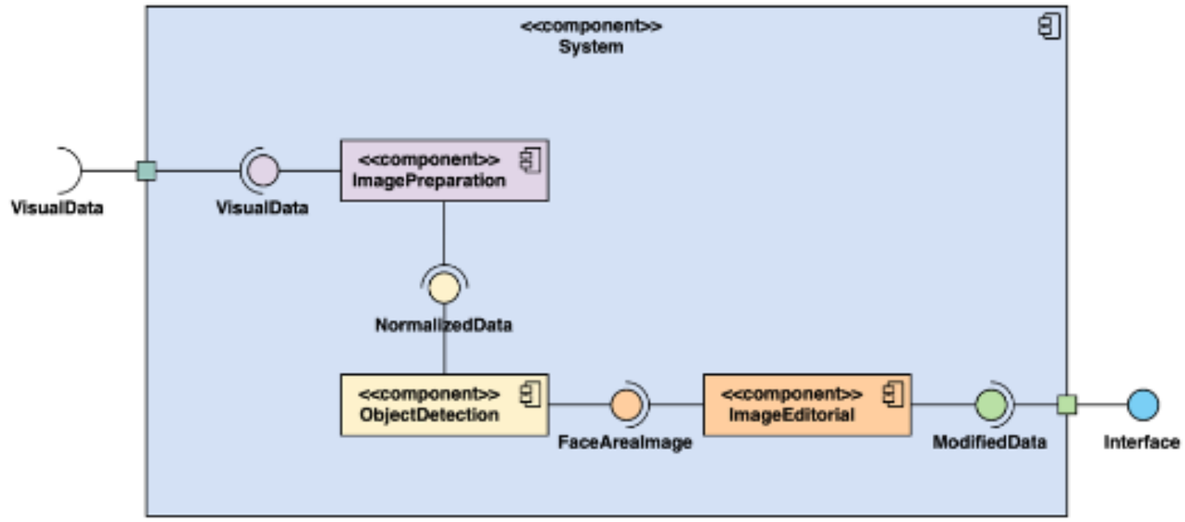
Зм.	Арк.	№ документа	Підпис	Дата
Розробив		Нго Май Фионг		
Перевірив		Ковалюк Т.В.		
Т. кон.				
Н. кон.		Халус О.А.		
Затвердив		Ковалюк Т.В.		

Схема структурна класів програмного забезпечення

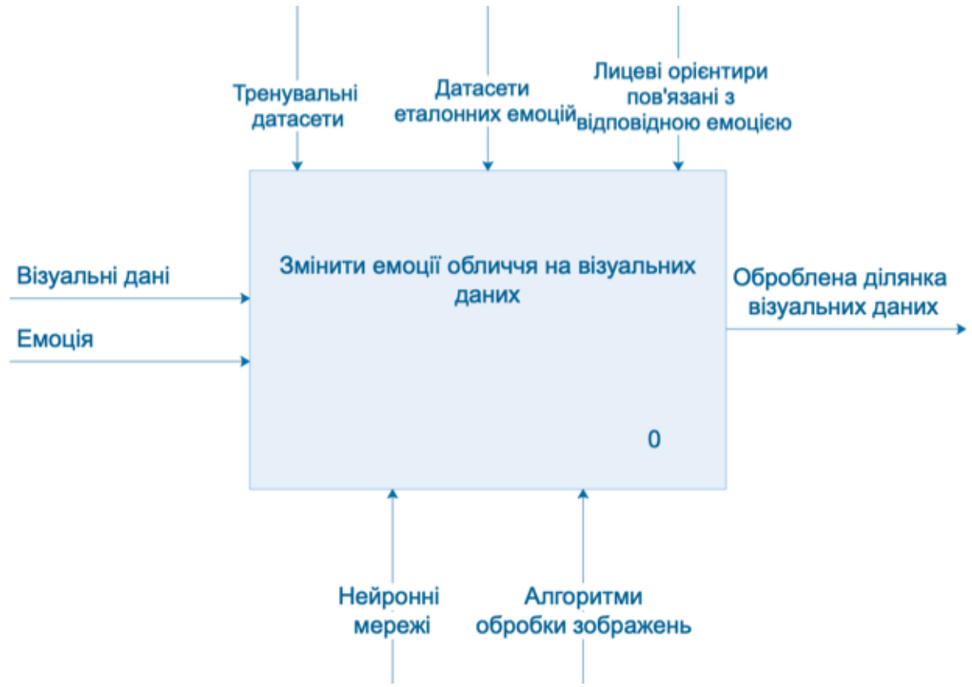
Літера	Маса	Масштаб
Аркуш 1	Аркушів 1	

Система обробки зображень облич людей за допомогою алгоритмів комп'ютерного зору

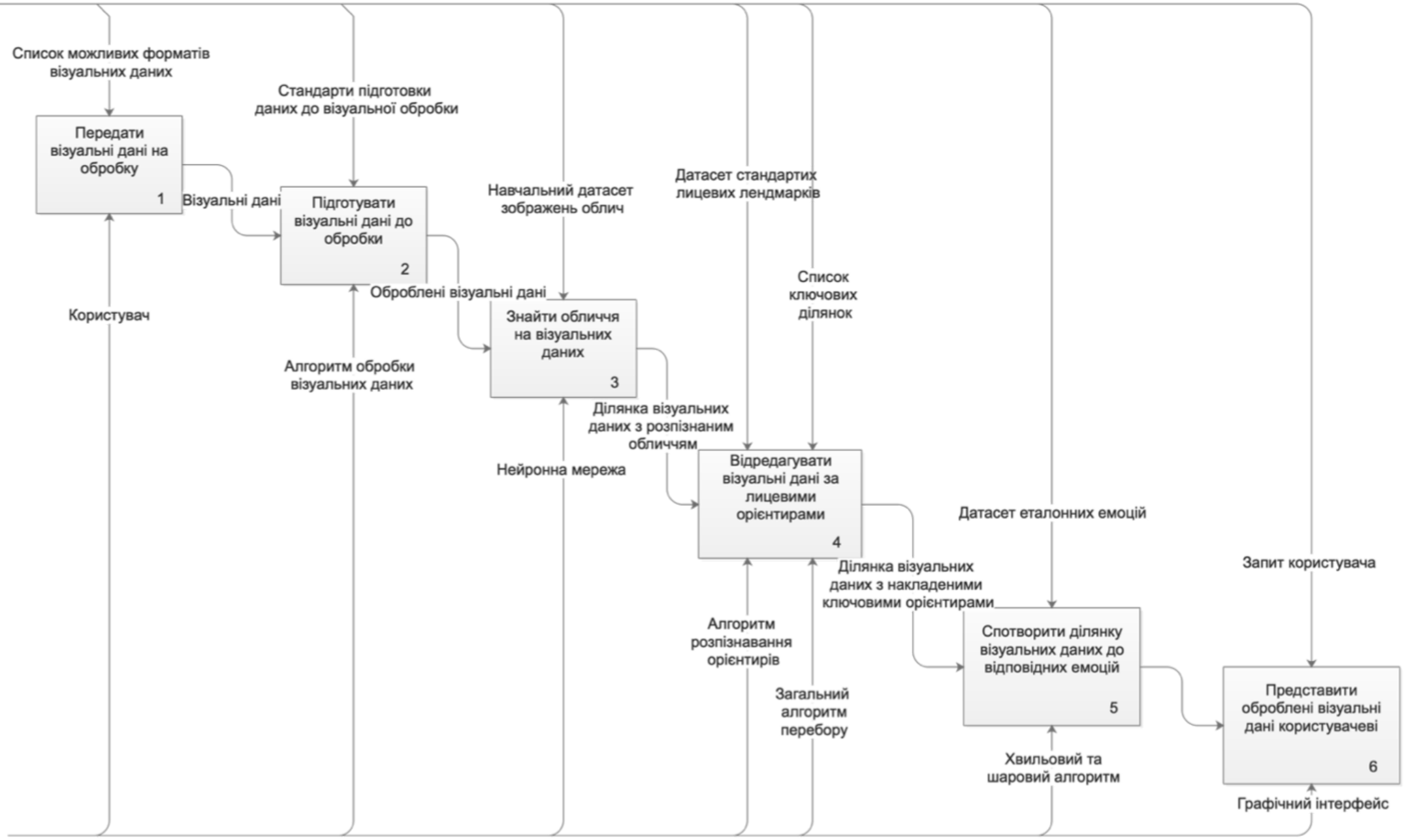
КПІ ім. Сікорського ФІОТ
Кафедра АСОІУ
гр. ІС-52



					ДП ІС-5218.1181-с.СКС
Зм.	Арк.	№ документа	Підпис	Дата	*Схема структурна компонентів системи*
Розробив		Нго Май Фионг			Літера: — Маса: — Масштаб: —
Перевірив		Ковалюк Т.В.			Аркуш 1 / Аркушів 1
Т. кон.					Система обробки зображень облич людей за допомогою алгоритмів комп'ютерного зору
Н. кон.		Халус О.А.			КПІ ім. Сікорського ФІОТ Кафедра АСОІУ гр. ІС-52
Затвердив		Ковалюк Т.В.			

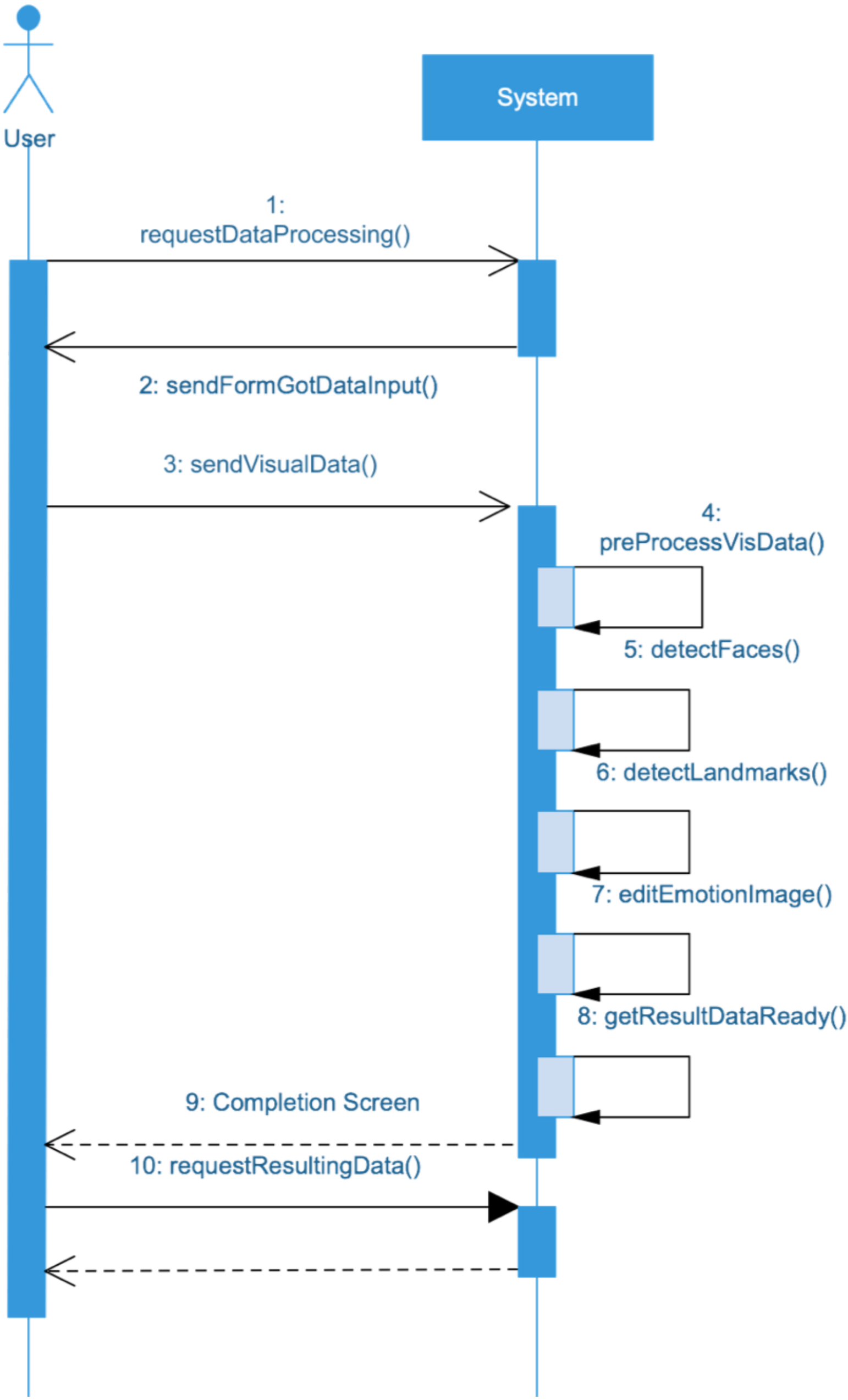


					ДП ІС-5218.1181-с.СКМ	
Зм.	Арк.	№ документа	Підпис	Дата	*Схема структурна контекстної моделі системи*	Літера / Маса / Масштаб
Розробив		Нго Май Фионг				
Перевірив		Ковалюк Т.В.				Аркуш 1 / Аркушів 1
Т. кон.						
					Система обробки зображень облич людей за допомогою алгоритмів комп'ютерного зору	КПІ ім. Сікорського ФІОТ Кафедра АСОІУ гр. ІС-52
Н. кон.		Халус О.А.				
Затвердив		Ковалюк Т.В.				

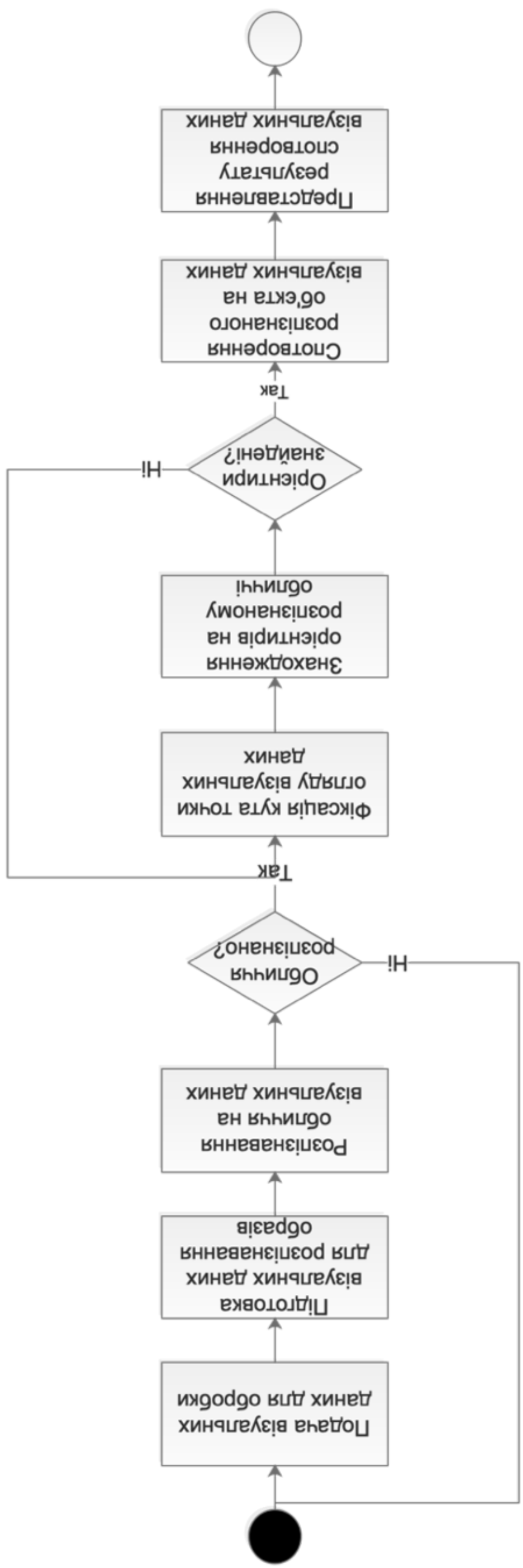


					ДП ІС-5218.1181-с.СФМ
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна функціональної моделі IDEF0
Розробив		Нго Май Фионг			Літера: — Маса: — Масштаб: —
Перевірив		Ковалюк Т.В.			Аркуш 1 / Аркушів 1
Т. кон.					Система обробки зображень облич людей за допомогою алгоритмів комп'ютерного зору
Н. кон.		Халус О.А.			КПІ ім. Сікорського ФІОТ Кафедра АСОІУ гр. ІС-52
Затвердив		Ковалюк Т.В.			

ДП ІС-5218.1181-с.СПС



					ДП ІС-5218.1181-с.СПС	
Зм.	Арк.	№ докум.	Підп.	Дата	Схема структурна послідовності системи	Лит. / Маса / Масштаб
Розроб.		Нго Май Фионг				
Перев.		Ковалюк Т.В.				Аркуш 1 / Аркушів 1
Т. Кон.					Система обробки зображень облич людей за допомогою алгоритмів комп'ютерного зору	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52
Н. Кон.		Халус О.А.				
Затв.		Ковалюк Т.В.				



Зм.	Арк.	№ докум.	Підп.	Дата
Розроб.		Нго Май Фионг		
Перев.		Коваленок Т.В.		
Т. контр.				
Н. контр.		Харчук О.А.		
Затв.		Коваленок Т.В.		

ДП ІС-5218.1181-с.ССД

Інв. № підл.	Підп. і дата	Взам. інв. №
Інв. № дубл.	Підп. і дата	

Лист	Маса	Масштаб		
Аркуш 1		Аркушів 1		

Схема структурна діяльності системи

Система обробки зображень облич людей за допомогою алгоритмів комп'ютерного зору

КПІ ім. Ігоря Сікорського
кафедра АСОІУ гр. ІС-52

ДП ІС-5218.1881-с.ССД